UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

MÉMOIRE D'ÉTUDIANT

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

CRT-622

APPLICATION D'UNE MÉTHODE DE "BRANCH-AND BOUND" AU PROBLÈME DE LOCALISATION AVEC TRANSFERTS ENTRE LES DÉPÔTS

par

Jacques Hodgson jr.

Mémoire de maîtrise, département d'Informatique et de Recherche opérationnelle, dirigé par Teodor G. Crainic.

Université de Montréal

Centre de recherche sur les transports - Publication #622

Décembre 1988

Dépôt légal - 1er trimestre 1989

SOMMAIRE

Le problème de localisation avec transferts entre les dépôts consiste à minimiser les coûts totaux d'une compagnie de transport qui reçoit des véhicules vides de ses clients à ses dépôts et qui les retourne à d'autres clients pour rechargement.Ceci implique donc de trouver les meilleurs endroits possibles pour installer ces dépôts. De plus, il peut exister plusieurs types de véhicules. Dans ce mémoire, ce problème est appelé problème de localisation avec équilibrage.

Un équilibrage est nécessaire pour que chaque dépôt reçoive exactement le même nombre de véhicules qu'il expédie, ceci pour chaque type. Puisque dans une solution optimale, chaque dépôt ne reçoit pas nécessairement des clients exactement ce qu'il doit expédier, les dépôts recevant un surplus de véhicules doivent en envoyer à ceux qui n'en reçoivent pas assez. Des mouvements de véhicules sont donc permis entre deux dépôts.

Un modèle de programmation linéaire mixte a été développé pour ce problème. Les variables entières de cette formulation correspondent à l'état des dépôts qui seront soit ouverts, soit fermés. Plusieurs méthodes de résolution ont été envisagées et constituent divers projets de maîtrise. Dans ce mémoire nous analysons une méthode d'énumération ("branch-and-bound").

La formulation complète de la localisation avec équilibrage étant complexe à résoudre, nous appliquons notre méthode d'énumération à une relaxation faible qui possède des propriétés intéressantes. Nous obtenons alors un problème séparable par types de véhicules, ayant une structure de réseau. Ces nouveaux sous-problèmes

représentent des problèmes de flot à coût minimum sans capacité.

Ce problème ressemble au problème de localisation simple, bien que dans ce dernier aucun mouvement direct ne soit permis entre deux dépôts. Khumawala (1972) utilise un algorithme d'énumération sur une relaxation faible de ce nouveau problème et obtient des résultats intéressants.

Khumawala propose plus particulièrement des méthodes de sélection des dépôts devant être fixés ouverts ou fermés. Nous avons donc adapté certaines de ses méthodes au problème de localisation avec équilibrage. Nous développons également d'autres méthodes en tenant compte de la structure particulière du problème.

Le but de notre expérimentation est de déterminer l'efficacité de chacune des méthodes de sélection. Deux critères de comparaisons des méthodes sont retenus. Ces deux critères sont le temps d'exécution de la méthode et le nombre d'itérations nécessaires pour découvrir la solution optimale. Nous examinons également la possibilité de combiner plusieurs méthodes.

Nous testons également notre algorithme sur des données réelles provenant d'une compagnie de transport par conteneurs. Ces tests nous permettent de comparer l'efficacité de notre modèle mathématique par rapport aux résultats de cette compagnie.

Nous avons programmé en FORTRAN un algorithme d'énumération qui nous permet de tester les différentes méthodes de sélection des dépôts. Le programme développé utilise le progiciel RNET pour la résolution des sous-problèmes de flot à coût minimum sans capacité. L'efficacité de RNET nous permet d'obtenir un algorithme suffisamment rapide.

Nous comparons donc ces différentes méthodes de sélection, ce qui nous permet d'en former une nouvelle à partir d'une combinaison des deux meilleures méthodes. Cette nouvelle méthode, nous a permis d'améliorer les performances générales de l'algorithme. Nous avons testé notre algorithme sur un problème de grande dimension provenant d'une compagnie de transport par conteneurs. Nous avons obtenu un temps de résolution d'environ une demi journée pour ce problème, ceci sur un micro-ordinateur compatible PC AT muni d'une carte Definicon DSI-32 offrant un coprocesseur 32 bits avec une mémoire de deux méga-octets. Nous avons obtenu une économie de 41% sur les coûts totaux de la compagnie. Ceci représente une économie théorique puisque notre modèle se situe au niveau tactique. Le résultat le plus intéressant provient du fait que dans notre solution optimale 48 dépôts sont utilisés contre 87 dans les opérations régulières de la compagnie.

Mots-clés : localisation, équilibrage interdépôts, programmation mixte, méthodes d'énumération.

TABLE DES MATIERES

Sommaire iii

Table des matières vi

Liste des tableaux viii

Liste des figures ix

Remerciements x

Introduction 1

1. Modélisation 3

1.1 Problème de localisation avec équilibrage 3

1.2 Les problèmes de flot à coût minimum 9

1.3 Méthode de résolution 12

2. La localisation simple 26

2.1 Définition 26

2.2 Formulations 26

2.3 La relaxation faible 36

2.4 Méthode de Khumawala 41

2.5 Conclusion 52

3. La localisation avec équilibrage 54

3.1 Formulation 54

3.2 Le "branch-and-bound" 65

3.3 Conclusion 84

4. Les résultats 85

4.1 Problèmes tests 85

4.2 Implantation 87

4.3 Méthodes de sélection de variables 88

4.4 Tests sur la nouvelle règle 91

4.5 Tests sur données réelles 96

Conclusions 98

Références xi

Annexe A – Guide d'utilisation du programme de "branch-and-bound" sur la localisation avec transferts entre les dépôts xiii

Annexe B – Programme de "branch-and-bound" sur la localisation avec transferts entre les dépôts xxvi

LISTE DES TABLEAUX

Tableau 1 Caractéristiques des problèmes générés 86

Tableau 2 Comparaisons entre les méthodes de sélection 89

Tableau 3 Résultats sur la nouvelle méthode 92

Tableau 4 Autres résultats sur MZ1P1 94

Tableau 5 Comparaisons entre la nouvelle méthode et
les autres retenues comme efficaces 95

Tableau 6 Tests sur données réelles 97

LISTE DES FIGURES

I Structure du problème de localisation avec équilibrage 5

II Structure d'arbre d'un problème de "branch-and-bound" 17

REMERCIEMENTS

Je désire remercier mon directeur de recherche Teodor Gabriel Crainic pour m'avoir présenté ce sujet de thèse et également son appui tout au long de ma maîtrise.

Je veux également remercier Louis Delorme pour son aide depuis le début. Je tiens aussi à le remercier pour son support dans les moments difficiles.

Je remercie le Centre de Recherche sur les Transports (CRT) pour l'utilisation d'un de ses micro-ordinateurs.

Finalement, je tiens à remercier le CRSNG pour la bourse d'études supérieures qui m'a été accordée et sans laquelle la réalisation de ce travail aurait été impossible.

INTRODUCTION

Le problème de localisation avec transferts entre les dépôts consiste à minimiser les coûts totaux d'une compagnie de transport qui reçoit des véhicules vides de ses clients à ses dépôts et qui les retourne à d'autres clients pour rechargement. Ceci implique donc de trouver les meilleurs endroits possibles pour installer ces dépôts. De plus, il peut exister plusieurs types de véhicules.

Dans ce mémoire, ce problème est appelé problème de localisation avec équilibrage. Puisqu'il est nécessaire que chaque dépôt reçoive exactement le même nombre de véhicules de chaque type qu'il expédie, un équilibrage est effectué en transférant des véhicules entre les dépôts.

Un modèle de programmation linéaire mixte a été développé pour résoudre ce problème. Les variables entières de ce modèle représentent l'état des dépôts qui sont soit ouverts, soit fermés.

L'objectif de ce mémoire est de résoudre ce problème à l'aide d'un algorithme d'énumération ("branch-and-bound") basé sur une méthode semblable utilisée par Khumawala (1972) pour résoudre le problème de localisation simple. Nous testons plus particulièrement différentes méthodes de sélection des dépôts à ouvrir ou fermer, pour obtenir celle qui rendra l'algorithme le plus efficace. Nous testons également le modèle sur des données réelles provenant d'une compagnie de transport par conteneurs, pour comparer l'efficacité de notre solution avec celle de cette compagnie.

Dans le premier chapitre nous présentons le problème de localisation avec

équilibrage et certains modèles utilisés pour le résoudre. Nous présentons également la méthode du "branch-and-bound" appliquée à un problème linéaire de minimisation contenant des variables binaires.

Le chapitre suivant présente le problème de localisation simple et un algorithme pour le résoudre. L'étude de ce problème est intéressante puisqu'il ressemble au problème de localisation avec équilibrage.

Nous abordons ensuite la méthode de résolution par "branch-and-bound" du problème de localisation avec équilibrage. Ce chapitre est basé en grande partie sur Crainic, Dejax et Delorme (1986). Nous y présentons plus spécifiquement une relaxation du modèle mathématique permettant de résoudre le problème à l'aide d'un "branch-and-bound". Nous y développons également plusieurs méthodes de sélection des dépôts à ouvrir. De l'efficacité de ces méthodes va dépendre l'efficacité de l'algorithme du "branch-and-bound".

Nous terminons par des tests sur l'efficacité des différentes méthodes de sélection des dépôts,sur les performances générales de l'algorithme et sur la performance du modèle mathématique.

CHAPITRE 1 – MODELISATION

Dans ce chapitre nous présentons le problème de localisation avec équilibrage. Nous présentons également les problèmes de flot à coût minimum et de multiflot à coût minimum. Ces deux problèmes seront utilisés dans le troisième chapitre, pour résoudre le problème de localisation avec équilibrage.

Nous terminons par une présentation de l'algorithme du "branch-and-bound" adapté à un problème contenant des variables binaires.

1.1 – Problème de localisation avec équilibrage

La définition du problème de localisation avec équilibrage est tirée de Crainic, Dejax et Delorme (1986):

> Une entreprise de distribution ou transport livre à ses clients des marchandises à l'aide de véhicules de divers types. Pour chaque client l'entreprise connaît le nombre de véhicules pleins de chaque type à livrer. Lorsque reçus, les clients déchargent les véhicules, qui sont renvoyés à un dépôt. Les clients commandent aussi, pour leurs expéditions, des véhicules vides aux dépôts. Ces demandes sont également connues de l'entreprise. Une fois chargés, les véhicules sont transportés vers leur destination, normalement un autre point du réseau de l'entreprise, directement ou via leur dépôt d'origine ou un autre. Le transport peut s'effectuer par des modes (rail,

route, mode combiné, etc.) dont les caractéristiques de coût, de temps de transport, de fiabilité, etc. sont connues. Une complication majeure est dûe au fait que les demandes des clients entraînent un déséquilibre dans les disponibilités de véhicules vides entre les divers points du réseau de l'entreprise : certaines zones manquent de certains types de véhicules, tandis que des surplus sont enregistrés ailleurs. Ceci impose des transferts de véhicules vides entre les dépôts aux fins d'équilibrage. Il faut donc localiser les dépôts des véhicules vides pour satisfaire les demandes et minimiser le coût total de structure, de fonctionnement et de transport clients-dépôts, dépôts-clients et inter-dépôts.

La figure I représente le problème de localisation avec équilibrage. Dans ce problème, divers types de véhicules peuvent être utilisés. Ces types peuvent représenter des dimensions de véhicules (par exemple une remorque de 20 ou 40 pieds) ou des véhicules spécialisés tel des conteneurs réfrigérés, par exemple. Il ne faut pas confondre ce terme avec celui de mode qui représente plutôt la façon de transporter la marchandise (par exemple, modes routier, ferroviaire, etc.). Le terme produit est souvent utilisé à la place du terme type de véhicule, puisque généralement divers types de véhicules transportent des classes de produits différents.

Notons également que les offres et les demandes sont connues pour chaque produit et chaque client. De plus aucun mouvement direct d'un client à un autre n'est permis.

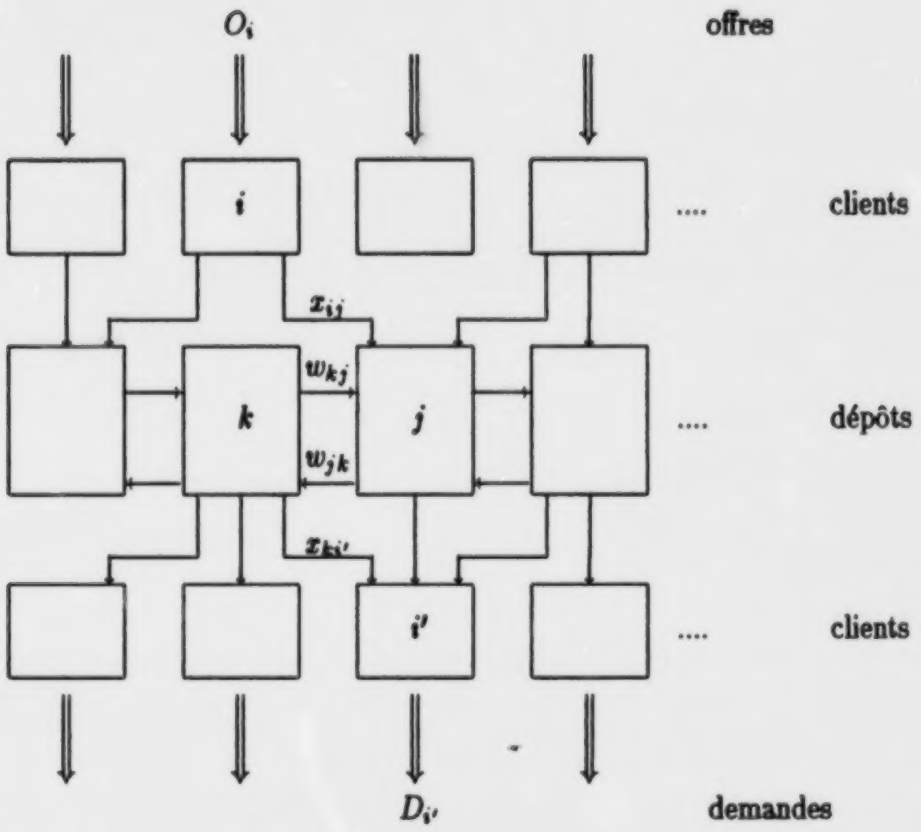

Figure I Structure du problème de localisation avec équilibrage

Finalement, les endroits où installer des dépôts seront choisis parmi un ensemble fini de localisations possibles. Ces localisations seront appelées dépôts candidats.

Définissons:

$C = \{ i \mid \text{client} \}$

$n = |C|$ = nombre de clients

$D = \{ j \mid \text{dépôt candidat} \}$

$m = |D|$ = nombre de dépôts candidats

$P = \{ p \mid \text{type de véhicule - produit} \}$

$l = |P|$ = nombre de produits

$T = \{ t \mid$ mode de transport $\}$

Ces éléments forment un réseau où C et D représentent les noeuds. Les arcs représentent les mouvements possibles pour chaque mode et chaque produit entre les dépôts, des dépôts vers les clients et vice versa. Donc entre deux noeuds, plusieurs arcs paralèles représentant chacun une combinaison de mode et de produit différente peuvent exister. Le réseau peut être défini comme étant:

$$G = (N, A)$$

où $\quad N = C \cup D$

$$A \subseteq (C \times D \times P \times T) \cup (D \times D \times P \times T) \cup (D \times C \times P \times T)$$

Définissons aussi:

O_{ip} : offre du client i pour le produit p, $i \in C$, $p \in P$;

D_{ip} : demande du client i pour le produit p, $i \in C$, $p \in P$;

$y = (y_j)$: vecteur d'ouverture des dépôts

$$y_j = \begin{cases} 1 & \text{dépôt } j \text{ ouvert} \\ 0 & \text{dépôt } j \text{ fermé} \end{cases} \qquad j \in D$$

$x = (x_{ijpt}, x_{jipt})$: vecteur des flots du produit p circulant par le mode t entre le client i et le dépôt j, $i \in C$, $j \in D$, $p \in P$, $t \in T$;

$w = (w_{jkpt})$: vecteur des flots du produit p circulant par le mode t entre les dépôts j et k, $j, k \in D$, $p \in P$, $t \in T$;

$f_j(y, x, w)$: fonction du coût d'ouverture et d'opération du dépôt j, $j \in D$;

$c_{ijpt}(x)$, $c_{jipt}(x)$: fonctions de coût de transport du produit p par le mode t entre le client i et le dépôt j, $i \in C$, $j \in D$, $p \in P$, $t \in T$;

$s_{jkpt}(w)$: fonction de coût de transport du produit p, par le mode t entre les dépôts j et k, $j, k \in D$, $p \in P$, $t \in T$.

Dans le cadre de ce mémoire nous étudierons seulement le cas où les fonctions de coût sont linéaires et non-négatives. Des fonctions concaves par exemple, pourraient être utilisées, pour tenir compte d'économies d'échelles réalisées lors du transfert des marchandises.

Puisque les fonctions de coût sont linéaires, lorsque un véhicule de type quelconque doit être transporté entre deux noeuds, le mode le moins couteux sera utilisé, peu importe la quantité de véhicule passant sur cet arc. Les autres modes permettant cette liaison sont alors inutiles. Le problème de localisation avec équilibrage peut être formulé ainsi: (Crainic, Dejax, Delorme (1986))

$$\text{(P1)} \quad \text{Min } Z = \sum_{j \in D} f_j y_j + \sum_{p \in P} \left\{ \sum_{i \in C} \sum_{j \in D} (c_{ijp} x_{ijp} + c_{jip} x_{jip}) + \sum_{j \in D} \sum_{k \in D} s_{jkp} w_{jkp} \right\}$$

$$\text{sujet à } \sum_{j \in D} x_{ijp} = O_{ip}$$

$$\sum_{j \in D} x_{jip} = D_{ip} \qquad \text{pour tout } i \in C, p \in P \qquad \text{(P1.1)}$$

$$x_{ijp} \leq M y_j$$

$$x_{jip} \leq M y_j \qquad \text{pour tout } i \in C, j \in D, p \in P \qquad \text{(P1.2)}$$

$$w_{jkp} \leq M y_j$$

$$w_{kjp} \leq My_j \qquad \text{pour tout } j,k \in D, p \in P \qquad \text{(P1.3)}$$

$$\sum_{i \in C} x_{ijp} + \sum_{k \in D} w_{kjp} - \sum_{i \in C} x_{jip} - \sum_{k \in D} w_{jkp} = 0$$
$$\text{pour tout } j \in D, p \in P \qquad \text{(P1.4)}$$

$$\begin{aligned} x_{ijp} &\geq 0 \\ x_{jip} &\geq 0 \qquad \text{pour tout } i \in C, j \in D, p \in P \\ w_{jkp} &\geq 0 \qquad \text{pour tout } j,k \in D, p \in P \\ y_j &\in \{0,1\} \qquad \text{pour tout } j \in D \end{aligned} \qquad \text{(P1.5)}$$

La fonction économique est composée de trois parties. La première tient compte des coûts d'ouverture des dépôts. La deuxième tient compte des coûts de transport entre les clients et les dépôts. La troisième partie tient compte des coûts de transport inter-dépôts.

Les contraintes de type (P1.1) assurent que les offres et les demandes de chaque client sont satisfaites. Les contraintes (P1.2) assurent que seulement les dépôts ouverts peuvent envoyer ou recevoir des véhicules des clients. M est une constante suffisamment grande pour assurer que toute l'offre et toute la demande de chaque client puisse passer par n'importe quel dépôt ouvert. Les contraintes (P1.3) assurent que les échanges de véhicules s'effectuent seulement entre dépôts ouverts. Les contraintes de type (P1.4) assurent la conservation de flot à chaque dépôt, c'est-à-dire qu'il entre exactement autant de véhicules vides qu'il en ressort. Finalement, les contraintes de type (P1.5) assurent que y_j est soit 0 (le dépôt j est fermé) soit 1 (le dépôt j est ouvert). Ces contraintes sont communément appelées contraintes

d'intégralité, parce qu'elles forcent les variables concernées à être entières. Les autres contraintes de ce type assurent simplement que les autres variables soient non-négatives.

1.2 – Les problèmes de flot à coût minimum

Un problème de flot à coût minimum est un problème de transport consistant à envoyer des marchandises (unités de flot) de certains endroits (sources) vers d'autres (puits), ceci à un coût minimum.

Dans la figure I, en considérant les usines comme des sources et les clients comme des puits, nous obtenons la configuration d'un problème de flot à coût minimum. Les dépôts représentent des noeuds de transit.

Chaque arc possède un coût de transport pour une unité de flot. Les arcs peuvent également posséder une capacité maximale sur le flot pouvant y circuler. Un problème de flot à coût minimum sans capacité est obtenu lorsque la capacité de chaque arc est infinie.

Lorsque plusieurs types de véhicules (produits) sont utilisés, nous obtenons un problème de multiflot à coût minimum sans capacité. Dans le troisième chapitre, nous verrons qu'une relaxation du problème de localisation avec équilibrage deviendra un problème de multiflot à coût minimal sans capacité.

Nous présentons maintenant les modèles mathématiques des problèmes de flot et de multiflot à coût minimum. Ces modèles seront utilisés dans le troisième chapitre pour résoudre le problème de localisation avec équilibrage. Nous présentons ces deux problèmes dans un contexte général.

Définissons d'abord un réseau $G = (N, A)$

où N représente les noeuds et A les arcs.

Les noeuds sont divisés en trois sous-ensembles:

N_S : ensemble contenant les noeuds sources;

N_P : ensemble contenant les noeuds puits;

N_T : ensemble contenant les autres noeuds (noeuds de transferts).

Notons que les sources et les puits peuvent également être utilisés comme noeuds de transferts.

Définissons également

O_r: offre de la source r pour tout $r \in N_S$;

D_r: demande du puits r pour tout $r \in N_P$;

a_{rs}: coût de transport unitaire sur l'arc allant de r à s, pour tout $r, s \in N$;

z_{rs}: nombre d'unités de flot passant sur l'arc allant de r à s, pour tout $r, s \in N$.

Le modèle est alors:

(FCM) Min $Z = \sum_{r \in N} \sum_{s \in N} a_{rs} z_{rs}$

sujet à $\sum_{s \in N} (z_{rs} - z_{sr}) = O_r$ pour tout $r \in N_S$ (FCM.1)

$\sum_{s \in N} (z_{rs} - z_{sr}) = 0$ pour tout $r \in N_T$ (FCM.2)

$\sum_{s \in N} (z_{rs} - z_{sr}) = -D_r$ pour tout $r \in N_P$ (FCM.3)

$z_{rs} \geq 0$ pour tout $r, s \in N$ (FCM.4)

La fonction économique de ce problème calcule simplement les coûts totaux de transport. Les contraintes de type (FCM.1) assurent que la différence entre les flots sortant et les flots entrant d'une source soit égale à l'offre de cette source. Ainsi, l'offre de cette source est ainsi entrée dans le réseau. Les contraintes (FCM.2) assurent que tout ce qui vient des autres noeuds dans ce noeud est égal à tout ce qui en sort. Les contraintes (FCM.3) assurent que la somme de tout ce qui sort, à un puits, moins, tout ce qui y entre, soit égale à moins la demande de ce puits. Ainsi, il restera à ce noeud le flot nécessaire pour couvrir sa demande.

Nous présentons maintenant le modèle mathématique (MCM) associé au problème de multiflot à coût minimum sans capacité.

Définissons l'ensemble P comme étant l'ensemble contenant les différents produits à transporter.

Définissons également

O_{rp} : offre de la source r pour le produit p,pour tout $r \in N_S$, $p \in P$;

D_{rp} : demande du puits r pour le produit p, pour tout $r \in N_P$, $p \in P$;

a_{rsp} : coût de transport unitaire sur l'arc allant de r à s pour le produit p, pour tout $r, s \in N$, $p \in P$;

z_{rsp} : nombre d'unités de flot passant sur l'arc allant de r à s pour le produit p, pour tout $r, s \in N$, $p \in P$.

Le modèle est alors

$$\text{(MCM)} \quad \text{Min } Z = \sum_{p \in P} \left\{ \sum_{r \in N} \sum_{s \in N} a_{rsp} z_{rsp} \right\}$$

$$\text{sujet à} \quad \sum_{s \in N} (z_{rsp} - z_{srp}) = O_{rp} \qquad \text{pour tout } r \in N_S,\ p \in P \qquad \text{(MCM.1)}$$

$$\sum_{s \in N} (z_{rsp} - z_{srp}) = 0 \qquad \text{pour tout } r \in N_T,\ p \in P \qquad \text{(MCM.2)}$$

$$\sum_{s \in N} (z_{rsp} - z_{srp}) = -D_{rp} \qquad \text{pour tout } r \in N_P,\ p \in P \qquad \text{(MCM.3)}$$

$$z_{rsp} \geq 0 \qquad \text{pour tout } r, s \in N,\ p \in P \qquad \text{(MCM.4)}$$

Nous voyons que ce modèle peut être séparé par produits puisque aucune contrainte n'affecte plusieurs produits. Donc les contraintes peuvent être regroupées par produit. De plus la fonction économique est séparable en fonctions, chacune sur un produit en particulier.

1.3 – Méthode de résolution

Nous abordons ici notre méthode de résolution. La méthode du "branch-and-bound" est utilisée lorsque dans un problème de programmation mathématique certaines variables sont entières. D'autres méthodes pourraient être plus rapides, dépendant de la structure du problème. La méthode du "branch-and-bound" possède cependant l'avantage de converger vers la solution optimale et non seulement une solution près de celle-ci.

Mais en pratique l'algorithme prend fin avant d'obtenir l'assurance que la solution optimale a été atteinte. Ceci en raison du temps considérable nécessaire à l'exécution complète de la méthode. A ce moment la meilleure solution entière trouvée est retenue comme la meilleure solution. Il est donc important de se diriger rapidement vers la solution optimale ou une solution près de celle-ci. Ceci peut être obtenu en définissant des méthodes de sélection des variables entières adaptées à la

structure du problème à traiter.

Nous définissons également la relaxation d'un problème de programmation mathématique, puisqu'une relaxation du problème de localisation avec rééquilibrage sera utilisée dans notre méthode de résolution. Ceci est nécessaire en raison du nombre trop élevé de contraintes, particulièrement celles de types (P1.2) et (P1.3).

1.3.1 – La relaxation

Définissons d'abord la relaxation d'un problème de programmation mathématique. Un problème est la relaxation d'un autre si son ensemble solution contient l'ensemble solution du deuxième. En définissant

$E1$ et $E2$: ensembles solutions de deux problèmes de programmation linéaire $PL1$ et $PL2$

alors $PL1$ est une relaxation de $PL2$ si et seulement si

$$E2 \subseteq E1,$$

et $PL1$ et $PL2$ possèdent des expressions mathématiques équivalentes comme fonction économique.

En pratique, on obtient une relaxation en remplaçant certaines contraintes d'un problème par d'autres, moins restrictives, ou par élimination de ces contraintes.

Proposition 1.1 : (Salkin (1975)) Si $PL1$ est une relaxation de $PL2$, alors la valeur de la solution optimale de $PL1$ est plus petite ou égale à la valeur de la solution optimale de $PL2$ (ceci pour un problème de minimisation).

Preuve : (Salkin (1975)) Nous savons, par la définition d'une relaxation, que $E2 \subseteq E1$ et que $PL1$ et $PL2$ ont la même fonction économique.

Soit $S2$ une solution optimale de $PL2$, alors $S2 \in E2$, et donc $S2 \in E1$.

Soit $S1$ une solution optimale de $PL1$, alors $S1 \in E1$ et puisque $S1$ est optimale alors $S1 \leq S2$.

c.q.f.d.

1.3.2 – Le "branch-and-bound"

Le "branch-and-bound" est une technique de résolution pour un problème de programmation mathématique contenant des variables entières. Dans le problème de localisation avec rééquilibrage, les variables entières sont les y_j qui peuvent prendre les valeurs 0 ou 1. Ces variables sont dites binaires puisqu'elles n'ont que deux valeurs possibles. Nous étudierons donc le "branch-and-bound" pour des variables binaires et pour un problème de minimisation.

Cette section sera divisée en six sous-sections. En premier lieu, nous ferons une présentation générale du "branch-and-bound". La deuxième sous-section traitera des bornes inférieures calculées à chaque noeud. Ensuite, nous discuterons des bornes supérieures sur la solution optimale du problème original. Ces bornes peuvent réduire la taille du "branch-and-bound" et le temps d'exécution de la méthode. Puis, nous abordons les méthodes de sélection de variables à fixer. La cinquième sous-section indique la méthode d'exploration de l'arbre du "branch-and-bound". Finalement nous présentons l'algorithme général qui incorpore les éléments des sous-

sections précédentes.

1.3.2.1 – Présentation générale

Le "branch-and-bound" consiste à remplacer les contraintes, dans un problème de programmation mathématique, forçant certaines variables à être binaires, par des contraintes d'intervalles, contenant les deux valeurs possibles. Le nouveau problème obtenu est alors une relaxation du problème original puisque toutes les solutions de ce dernier sont contenues dans le nouveau problème.

En pratique les limites de l'intervalle seront ces deux valeurs possibles. Ceci pour réduire le plus possible la taille de l'ensemble solution du problème relaxé.

Une contrainte de type

$$x \in \{a, b\}$$

sera remplacée par une contrainte de type

$$a \leq x \leq b.$$

Pour la suite du texte définissons:

P_{or}: Problème original contenant des variables binaires.

E_{or}: Ensemble solution de P_{or}.

S_{or}: Une solution optimale de P_{or}.

P_{re}: Problème P_{or} relaxé, c'est-à-dire où les contraintes forçant les variables à être binaires, sont remplacées par des contraintes d'intervalle.

E_{re}: Ensemble solution de P_{re}.

S_{re}: Une solution optimale de P_{re}.

Par la proposition 1.1, puisque P_{re} est une relaxation de P_{or},

$$E_{or} \subseteq E_{re} \text{ et } S_{or} \in E_{re}$$

Donc $S_{re} \leq S_{or}$.

Définissons également une solution entière à un problème de programmation mathématique, comme étant une solution où toutes les contraintes forçant les variables à être entières ou binaires sont satisfaites. Notons qu'une solution entière de P_{re} est une solution réalisable de P_{or}. Remarquons également qu'une solution réalisable de P_{or} est forcément entière.

Pour trouver une solution réalisable à P_{or}, il suffit de remplacer toutes les variables binaires par une de leurs valeurs possibles (une telle solution est entière), puis de calculer une solution réalisable s'il en existe une. Par exemple pour le problème de localisation avec rééquilibrage, si tous les dépôts étaient fermés ($y_j = 0$ pour tout j), le flot ne pourrait circuler et il n'y aurait aucune solution réalisable.

Par contre si pour une combinaison donnée des variables binaires, une solution réalisable peut être calculée, alors il est possible de calculer la meilleure solution réalisable pour cette combinaison.

En répétant ce processus pour toutes les combinaisons possibles de variables binaires, et en choisissant la meilleure solution parmi celles calculées, une solution optimale de P_{or} est obtenue.

Le "branch-and-bound" intervient comme étant une méthode permettant d'obtenir systématiquement la meilleure de ces combinaisons. La technique consiste à obtenir deux sous-problèmes à partir du problème relaxé (P_{re}). Il s'agit de choisir une variable binaire et de la fixer dans P_{re} à une de ses valeurs pour un sous-problème et de fixer la variable à son autre valeur pour l'autre sous-problème.

Le même processus est répété pour chacun de ces deux nouveaux problèmes

avec une variable binaire n'ayant pas été fixée précédemment. Le processus est encore répété pour les nouveaux problèmes jusqu'à l'obtention d'une solution non-réalisable ou d'une solution réalisable pour le problème original, c'est-à-dire une solution entière. Cette technique peut être illustrée par un arbre (figure II).

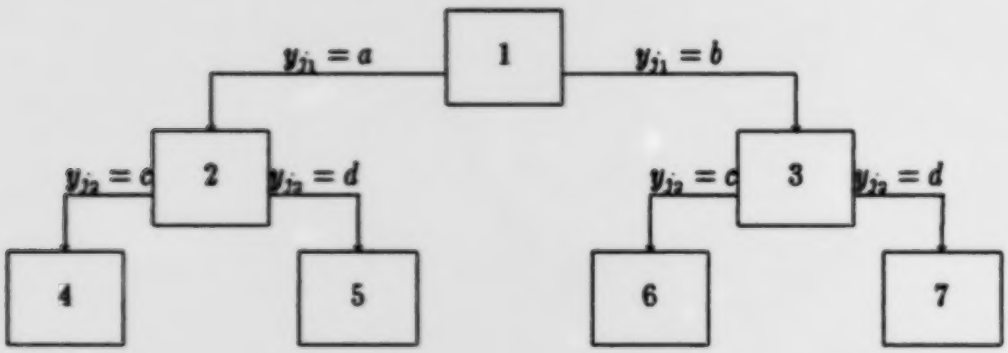

Figure II Structure d'arbre d'un problème de "branch-and-bound"

Cette structure d'arbre est dite binaire, puisque de chaque noeud, il sort exactement deux arcs ou branches. Les deux noeuds à l'extrémité de ces branches sont dits les enfants du premier noeud, et ce noeud est le parent des deux autres. Le noeud supérieur de l'arbre est appelé la racine. Notons qu'il existe un et un seul chemin allant de la racine à tout noeud de l'arbre. Les branches de ce chemin indiquent toutes les variables binaires qui ont été fixées pour atteindre ce noeud.

Dans cette structure les noeuds sont à divers niveaux. Deux noeuds sont sur le même niveau si le même nombre de variables binaires ont été fixées pour atteindre ces noeuds.

A chaque noeud de cet arbre, il faut remplacer dans P_{re} les variables déjà fixées dans le chemin venant de la racine, par leurs valeurs. Une solution optimale est

alors calculée pour ce nouveau problème.

Définissons maintenant le sous-arbre d'un noeud donné dans le "branch-and-bound" comme étant l'arbre composé des enfants de ce noeud, de ces petits enfants, etc., jusqu'au niveau le plus bas de l'arbre. Par exemple dans la figure II, le sous-arbre du noeud 2 est formé des noeuds 2, 4 et 5.

Définissons également l'exploration d'un noeud de l'arbre comme étant la suite d'opérations consistant à calculer la solution optimale à ce noeud, puis à choisir une nouvelle variable à fixer de façon à créer les deux noeuds enfants.

Proposition 1.2 : (Salkin (1975)) Un problème à un noeud donné du "branch-and-bound" est une relaxation de tout problème contenu dans son sous-arbre.

Preuve : (Salkin (1975)) Soit n_i un problème à un noeud quelconque du "branch-and-bound" et soit n_j un autre problème dont le noeud est situé dans le sous-arbre de n_i.

Alors n_j contient toutes les contraintes de n_i, mais en plus est limité par certaines variables qui ont été fixées. Ces variables fixées correspondent en fait à des contraintes supplémentaires sur le problème et puisqu'elles sont absentes dans n_i, par définition n_i est une relaxation de n_j.

c.q.f.d.

Proposition 1.3 : (Salkin (1975)) Un problème à un noeud donné du "branch-and-bound" possède toujours une solution optimale inférieure ou égale à tout noeud de son sous-arbre.

Preuve : (Salkin (1975)) Reprenons les n_i et n_j de la preuve précédente.

Soit

S_i : solution optimale à n_i.

S_j : solution optimale à n_j.

Puisque n_i est une relaxation de n_j alors $S_i \leq S_j$.

c.q.f.d.

Une conséquence de ce résultat est que si une solution entière est atteinte à un noeud donné du "branch-and-bound" avant que toutes les variables entières (ou binaires) aient été explicitement fixées, alors il est certain qu'aucune solution entière contenue dans le sous-arbre résultant ne pourra être meilleure. Il est donc inutile d'explorer ce sous-arbre.

1.3.2.2 – Bornes inférieures

Dans la présentation générale du "branch-and-bound", nous mentionnions qu'il était nécessaire de calculer à chaque noeud, la solution optimale de P_{re} dans lequel les variables binaires déjà fixées étaient remplacées par leurs valeurs. La valeur de la fonction économique de cette solution optimale constitue une borne inférieure sur toutes les solutions optimales des noeuds contenus dans le sous-arbre résultant, ceci par la proposition 1.3.

Si nous connaissons a priori une solution réalisable pour le problème original

et la valeur de sa fonction économique et si la valeur calculée à un noeud donné du "branch-and-bound" est plus grande que celle de la solution réalisable, il n'existera aucune solution réalisable pour le problème original meilleure que celle déjà connue dans le sous-arbre résultant. Dans ce cas il est inutile d'explorer ce sous-arbre. Dans la section suivante, nous verrons que la valeur de la fonction économique de cette solution réalisable est en fait une borne supérieure sur la solution optimale.

Notons finalement qu'à un noeud donné du "branch-and-bound", il n'est pas nécessaire de calculer la valeur de la fonction économique de P_{re}, mais qu'une borne inférieure sur cette dernière est suffisante. Ceci est possible par exemple en relaxant certaines contraintes de P_{re}.

1.3.2.3 – Borne supérieure

Dans cette section nous allons introduire le concept de borne supérieure sur la solution optimale de P_{or}. A l'aide de cette borne et de la proposition 1.3, nous pourrons éliminer certains noeuds de l'arbre du "branch-and-bound".

Supposons que la borne supérieure soit connue. Si à un noeud donné de l'arbre du "branch-and-bound" la borne inférieure calculée est plus grande que cette borne supérieure, alors il est inutile d'explorer le sous-arbre résultant de ce noeud. Ceci est dû à la proposition 1.3 qui indique que toutes les solutions du sous-arbre résultant seront supérieures ou égales à celle du noeud. Donc en particulier toutes les solutions entières contenues dans ce sous-arbre seront supérieures à la borne supérieure sur la solution optimale. Il est donc certain que la solution optimale de P_{or} ne se trouve pas dans ce sous-arbre.

Il reste à trouver cette borne supérieure. Une première façon est de fixer arbi-

trairement cette borne au début de l'algorithme. Il faut alors s'assurer qu'il s'agit vraiment d'une valeur supérieure à la solution optimale de P_{or}. Cette méthode peut quand même être utile car dépendant du problème, il peut être possible d'approximer une borne satisfaisante. Par exemple, après le calcul de la solution au premier noeud (la racine de l'arbre) du "branch-and-bound" (solution optimale de P_{re}), il est possible d'ajouter un certain pourcentage à cette solution et utiliser ce résultat comme borne.

Une deuxième méthode consiste simplement à prendre la meilleure solution trouvée jusqu'à présent, comme borne supérieure. La borne supérieure initiale est infinie.

Lorsqu'une solution entière, c'est-à-dire une solution réalisable pour P_{or}, est inférieure à la borne supérieure, cette solution devient la nouvelle borne supérieure.

1.3.2.4 – Méthodes de sélection des variables à fixer

Des méthodes de sélection des variables à fixer doivent également être développées. Ces méthodes dépendent généralement de la structure du problème de programmation mathématique. Une telle méthode permettra, à un noeud donné du "branch-and-bound", de choisir la variable à fixer et de connaître la valeur à laquelle elle devrait d'abord être fixée.

De telles méthodes sont en général plus ou moins intuitives. Elles ne peuvent garantir que le bon choix soit toujours fait. En effet, une méthode qui pourrait toujours garantir le bon choix, rendrait inutile l'utilisation du "branch-and-bound", puisqu'il serait possible de déterminer directement les valeurs des variables binaires dans une solution optimale.

Cependant une méthode qui permettra un bon choix plus souvent qu'une autre, pourra conduire à une bonne solution entière plus rapidement en réduisant la borne supérieure et donc le nombre de noeuds à explorer dans l'arbre.

1.3.2.5 – Exploration de l'arbre du "branch-and-bound"

Une méthode d'exploration du "branch-and-bound" doit posséder trois qualités. Premièrement la méthode doit permettre systématiquement l'exploration de tous les noeuds de l'arbre. La méthode doit également utiliser un minimum d'espace mémoire. Finalement, la méthode doit permettre le plus possible la réduction rapide de la borne supérieure pour diminuer le nombre de noeuds à explorer dans l'arbre.

Lors de l'exploration d'un noeud deux nouveau noeuds sont créés, sauf dans trois cas particuliers. Ces cas surviennent lorsque le noeud contient soit une solution entière, soit une solution non-réalisable, ou lorsque la valeur de la borne inférieure calculée est supérieure à la borne supérieure.

Pour explorer complètement l'arbre, il s'agit d'insérer d'abord la racine dans une liste. En général un noeud est sorti de la liste et exploré. S'il est nécessaire de créer les noeuds enfants, ils sont placés en fin de liste. Cette étape est répétée jusqu'à ce que la liste soit vidée.

L'arbre sera donc totalement exploré peu importe le noeud sélectionné à chaque étape. Mais pour systématiser la méthode il suffit de choisir soit le premier, soit le dernier noeud de la liste.

Si le premier noeud est systématiquement choisi, la racine sera d'abord explorée, puis ses deux enfants. Pour chacun de ces deux noeuds, les enfants seront créés et placés en fin de liste. L'arbre est donc exploré par niveau, ce qui correspond

à une recherche en largeur. Il est donc nécessaire de réserver un espace mémoire suffisant pour conserver tous les noeuds du niveau qui en compte le plus.

Si le dernier noeud est systématiquement choisi, alors lors de l'exploration d'un noeud, si deux noeuds enfants sont créés, ils sont placés en fin de liste et un des deux deviendra automatiquement le prochain noeud exploré. Ainsi chaque fois qu'un noeud est exploré et qu'il est nécessaire de créer les enfants, un des deux devient le prochain exploré et l'autre reste dans la liste. Puisque les enfants sont toujours au niveau suivant de celui de leur parent et qu'il y a autant de niveaux que de variables entières, il y aura au plus autant de noeuds dans la liste qu'il y a de variables entières. Donc cette méthode demande un espace mémoire moins considérable que la précédente.

1.3.2.6 – Synthèse et algorithme

Le "branch-and-bound" est une technique consistant à relaxer les contraintes forçant certaines variables d'un problème à être entières (ou binaires). Ces variables sont ensuite fixées une à une jusqu'à l'obtention d'une solution entière. Ce processus peut être illustré par un arbre dans lequel une borne inférieure sur la valeur de la fonction économique du problème original est calculée à chaque noeud. Comme le nombre de noeuds est élevé, il est important de le réduire le plus possible.

L'algorithme est exécuté sur un problème de programmation mathématique dans lequel les variables y_j sont binaires.

$$y_j \in \{a_j, b_j\} \qquad \text{pour tout } j$$

Nous définissons également un ensemble L contenant les variables binaires fixées et leurs valeurs pour tout noeud de l'arbre. Nous avons également une pile de ces

éléments. De plus la variable B_{sup} représente la borne supérieure sur la meilleure solution entière du problème original.

L'algorithme contient également deux fonctions, $ENTRE(L)$ qui entre l'ensemble L dans la pile et $SORT(L)$ qui sort un élément de la pile et le place dans l'ensemble L.

Les étapes de l'algorithme sont :

1 - [Initialisations]

$L = \emptyset$ (le premier noeud est la racine n'ayant aucune variable fixée

La pile est vidée

$ENTRE(L)$

2 - [Sélection d'un noeud à résoudre]

si la pile est vide alors l'algorithme est terminé

sinon $SORT(L)$

3 - [Calcul de la borne inférieure]

Trouver une borne inférieure Z sur P_{re} avec les variables de l'ensemble L fixées.

4 - [Vérification de solution réalisable]

Si la solution calculée est non-réalisable pour le problème original

alors aller à 2

5 – [Test sur la borne inférieure]

Si $Z \geq B_{sup}$ alors aller à 2

6 – [Vérification de solution entière]

Si la borne inférieure Z représente une solution entière alors

Calcul de la valeur de cette solution Z' pour le problème original.

Si $Z' < B_{sup}$ alors

$B_{sup} = Z'$

Z' devient la nouvelle borne supérieure

aller à 2

7 – [Sélection d'une nouvelle variable binaire à fixer]

Choisir une variable binaire j n'appartenant pas à L

Ajouter j à L avec valeur a_j

$ENTRE(L)$

Ajouter j à L avec valeur b_j

$ENTRE(L)$

Aller à 2

CHAPITRE 2 – LA LOCALISATION SIMPLE

Dans ce chapitre, nous présentons le problème de localisation simple qui ressemble à celui de localisation avec équilibrage, bien que moins complexe que ce dernier. Comme le problème de localisation simple a été bien étudié, il est possible de s'inspirer de ces résultats pour résoudre le problème de localisation avec équilibrage.

Nous présentons d'abord la définition du problème, puis différentes formulations retrouvées dans la littérature. Le chapitre se termine par une section sur le "branch-and-bound" comme méthode de résolution pour une relaxation faible du problème. Nous nous attarderons surtout sur la méthode de Khumawala (1972).

2.1 – Définition

Le problème de localisation simple peut s'énoncer ainsi:

Une entreprise de distribution ou transport livre à ses clients des marchandises à partir de dépôts. Il s'agit de trouver les meilleurs endroits pour installer ces dépôts, de façon à minimiser les coûts totaux. Ces coûts totaux sont en fait la somme des coûts de transport et des coûts d'ouverture de ces dépôts.

2.2 – Formulations

Cette section est réservée à la présentation de différentes formulations retrou-

vées dans la littérature.

Le problème de localisation simple peut être formulé ainsi: (Krarup et Pruzan (1983))

$$\text{(P2)} \quad \text{Min } Z = \sum_{j \in D} \sum_{i \in C} c_{ji} x_{ji} + \sum_{j \in D} f_j y_j$$

$$\text{sujet à} \quad \sum_{j \in D} x_{ji} = D_i \qquad \text{pour tout } i \in C \qquad \text{(P2.1)}$$

$$x_{ji} \leq M y_j \qquad \text{pour tout } j \in D, i \in C \qquad \text{(P2.2)}$$

$$x_{ji} \geq 0 \qquad \text{pour tout } j \in D, i \in C \qquad \text{(P2.3)}$$

$$y_j \in \{0,1\} \qquad \text{pour tout } j \in D \qquad \text{(P2.4)}$$

Essayons d'estimer la constante M. Cette constante doit être suffisamment grande pour assurer que si un dépôt j est fixé ouvert, aucune restriction, sur le flot sortant de ce dépôt n'existera. Mais le flot sur un arc $j \rightarrow i$ est limité par la demande du client i. Donc en remplaçant M par le D_i correspondant dans chaque contrainte de type (P2.2), chaque x_{ji} pourra transporter si nécessaire toute la demande du client i (D_i), lorsque le dépôt j sera ouvert ($y_j = 1$). De plus, si le dépôt j est fermé ($y_j = 0$), les x_{ji} correspondants ne peuvent être que nuls.

Les contraintes (P2.2) deviennent donc:

$$x_{ji} \leq D_i y_j \qquad \text{pour tout } j \in D, i \in C \qquad \text{(P2.2)}$$

Cette formulation considère que les x_{ji} représentent le nombre de véhicules allant du dépôt j au client i. Mais ces x_{ji} peuvent aussi représenter la fraction de la demande du client i passant sur le lien de j à i. Cornuejols, Nemhauser et Wolsey(1983) proposent donc la formulation suivante où

$$X_{ji} = x_{ji}/D_i \iff x_{ji} = X_{ji}D_i \qquad :$$

$$\text{Min } Z = \sum_{j\in D}\sum_{i\in C} c_{ji}X_{ji}D_i + \sum_{j\in D} f_j y_j$$

$$\text{sujet à} \quad \sum_{j\in D} X_{ji}D_i = D_i \qquad \text{pour tout } i \in C$$

$$X_{ji}D_i \leq D_i y_j \qquad \text{pour tout } j \in D, i \in C$$

$$X_{ji}D_i \geq 0 \qquad \text{pour tout } j \in D, i \in C$$

$$y_j \in \{0,1\} \qquad \text{pour tout } j \in D$$

et en posant $C_{ji} = c_{ji}D_i$ nous obtenons

$$\text{(P3)} \quad \text{Min } Z = \sum_{j\in D}\sum_{i\in C} C_{ji}X_{ji} + \sum_{j\in D} f_j y_j$$

$$\text{sujet à} \quad \sum_{j\in D} X_{ji} = 1 \qquad \text{pour tout } i \in C \qquad \text{(P3.1)}$$

$$X_{ji} \leq y_j \qquad \text{pour tout } j \in D, i \in C \qquad \text{(P3.2)}$$

$$X_{ji} \geq 0 \qquad \text{pour tout } j \in D, i \in C \qquad \text{(P3.3)}$$

$$y_j \in \{0,1\} \qquad \text{pour tout } j \in D \qquad \text{(P3.4)}$$

Cette formulation differe légèrement de celle des auteurs mentionnés, puisque ceux-ci forcent également les X_{ji} à être des variables binaires. Ceci n'est pas contraignant puisque nous verrons par la proposition 2.2 qu'il existe toujours une solution optimale où les X_{ji} sont entiers.

Khumawala (1972) propose une autre formulation

$$\text{(P4)} \quad \text{Min } Z = \sum_{j \in D} \sum_{i \in P_j} C_{ji} X_{ji} + \sum_{j \in D} f_j y_j$$

$$\text{sujet à} \quad \sum_{j \in N_i} X_{ji} = 1 \qquad \text{pour tout } i \in C \qquad \text{(P4.1)}$$

$$\sum_{i \in P_j} X_{ji} \leq n_j y_j \qquad \text{pour tout } j \in D \qquad \text{(P4.2)}$$

$$X_{ji} \geq 0 \qquad \text{pour tout } j \in D, i \in C \qquad \text{(P4.3)}$$

$$y_j \in \{0,1\} \qquad \text{pour tout } j \in D \qquad \text{(P4.4)}$$

où $C_{ji} = c_{ji} D_i$.

D_i = demande du client i.

$X_{ji} = x_{ji}/D_i$.

N_i = ensemble des dépôts qui peuvent fournir le client i.

P_j = ensemble des clients qui peuvent être servis par le dépôt j.

n_j = nombre d'éléments dans P_j.

Une différence importante apparaît entre les deux formulations, puisque Khumawala introduit les ensembles N_i et P_j, alors que dans (P3), chaque client peut être desservi par n'importe quel dépôt. La formulation de Khumawala est plus précise, mais il est quand même possible de la modifier pour la rendre compatible avec la précédente. Pour cela, il suffit d'ajouter les C_{ji} manquants, c'est-à-dire ceux pour lesquels il n'y a pas de lien entre le dépôt j et le client i. Une valeur suffisamment grande est donnée à ces C_{ji} de façon à ne pas utiliser ces liens. La formulation de Khumawala devient alors:

$$\text{(P5)} \quad \text{Min } Z = \sum_{j \in D} \sum_{i \in C} C_{ji} X_{ji} + \sum_{j \in D} f_j y_j$$

$$\text{sujet à} \quad \sum_{j \in D} X_{ji} = 1 \qquad \text{pour tout } i \in C \qquad \text{(P5.1)}$$

$$\sum_{i \in C} X_{ji} \leq n y_j \qquad \text{pour tout } j \in D \qquad \text{(P5.2)}$$

$$X_{ji} \geq 0 \qquad \text{pour tout } j \in D, i \in C \qquad \text{(P5.3)}$$

$$y_j \in \{0,1\} \qquad \text{pour tout } j \in D \qquad \text{(P5.4)}$$

Cette formulation est compatible avec celle de (P3). Notons ici que $n_j = n$ pour tout $j \in D$, où n est le nombre de clients, ceci parce que tous les arcs manquants ont été rajoutés, donc chaque dépôt peut servir tous les clients.

Proposition 2.1 : (Krarup et Pruzan (1983)) Les problèmes (P3) et (P5) possèdent les mêmes solutions optimales.

Preuve : (Krarup et Pruzan (1983))

1- Les deux problèmes possèdent la même fonction économique.

2- L'ensemble solution d'un problème est déterminé seulement par ses contraintes.

Puisque seules les contraintes (P3.2) et (P5.2) diffèrent dans les deux problèmes, il s'agit de démontrer qu'elles sont équivalentes étant donné les autres contraintes.

A) Supposons que X'_{ji} et y'_j, pour tout $i \in C$ et $j \in D$, soit une solution réalisable de (P3), alors cette solution satisfait les contraintes (P3.2)

$$X'_{ji} \leq y'_j \qquad \text{pour tout } i \in C, j \in D$$

$$\sum_{i \in C} X'_{ji} \leq \sum_{i \in C} y'_j = n y'_j$$

Ce qui satisfait les contraintes de type (P5.2). Donc une solution réalisable de (P3) est également réalisable pour (P5).

B) Supposons maintenant que X'_{ji} et y'_j, pour tout $i \in C$ et $j \in D$ est une solution réalisable de (P5), alors cette solution satisfait les contraintes de type (P5.2)

$$\sum_{i \in C} X'_{ji} \leq n y'_j \qquad \text{pour tout } j \in D$$

a) Prenons les y'_j tel que $y'_j = 0$, alors

$$\sum_{i \in C} X'_{ji} \leq 0$$

et puisque selon les contraintes (P5.3), alors

$$X'_{ji} \geq 0 \qquad \text{pour tout } i \in C, j \in D$$

ce qui implique que

$$X'_{ji} = 0 = y'_j \qquad \text{pour tout } i \in C$$

qui satisfait les contraintes de type (P3.2).

b) Prenons les y'_j tel que $y'_j = 1$

Puisque selon les contraintes (P5.1)

$$\sum_{j \in D} X'_{ji} = 1 \qquad \text{pour tout } i \in C$$

et que selon les contraintes de type (P5.3)

$$X'_{ji} \geq 0 \qquad \text{pour tout } i \in C, j \in D$$

alors $X'_{ji} \leq 1 = y'_j \qquad$ pour tout $i \in C, j \in D$

ce qui correspond aux contraintes de type (P3.2)

Donc une solution réalisable de (P5) est également réalisable pour (P3).

Puisque les deux problèmes ont la même fonction économique et le même ensemble solution, ils auront également les mêmes solutions optimales.

c q.f.d.

Une autre constatation importante pour résoudre le problème de localisation simple est celle-ci:

Proposition 2.2 : (Effroymson et Ray (1966)) Dans une solution optimale du problème de la localisation, tout client peut être desservi complètement par le dépôt ouvert le plus rapproché.

Preuve : (Effroymson et Ray (1966))

Dans une solution optimale du problème de localisation simple

$y_j \in \{0,1\}$ pour tout $j \in D$,

pour les formulations (P3) et (P5).

Puisque les clients ne peuvent être servis que par les dépôts ouverts, supposons qu'il existe $i \in C$ tel que ce client est servi par deux dépôts ouverts y_{j_1} et y_{j_2}. Alors

$$y_{j_1} = y_{j_2} = 1 \qquad \text{(dépôts ouverts)}$$

$$\text{et } 0 < X_{j_1 i} < 1 \,,\; 0 < X_{j_2 i} < 1$$

Supposons sans perte de généralité que

$$C_{j_1 i} \leq C_{j_2 i}$$

alors $C_{j_1 i} X_{j_1 i} \leq C_{j_2 i} X_{j_2 i}$

et donc le flot allant de j_1 à i ne coûterait pas plus cher à transporter entre j_2 et i, donc la solution serait au moins aussi bonne si tous le flot venait de j_1.

c.q.f.d.

Ainsi, le problème de localisation simple possède la propriété d'affectation unique, c'est-à-dire qu'un client peut être desservi seulement par le dépôt le plus rapproché dans toute solution optimale.

Pour résoudre ces deux formulations ((P3) et (P5)) du problème de localisation simple par l'algorithme du "branch-and-bound", il est nécessaire dans les deux cas de relaxer les contraintes forçant les variables y_j à être binaires. Ces contraintes sont remplacées par des contraintes forçant simplement ces y_j à être non-négatifs. Ceci est différent de l'approche décrite dans la section sur le "branch-and-bound" du chapitre précédent, mais nous verrons dans la section suivante que des contraintes d'intervalle sur les y_j sont inutiles dans ce contexte particulier. Les deux problèmes obtenus sont dénotés (P3r) et (P5r). Cependant ces deux nouveaux problèmes ne possèdent plus nécessairement le même ensemble de solutions réalisables.

Proposition 2.3 : (Krarup et Pruzan (1983))

1-) (P3r) et (P5r) ne possèdent pas nécessairement le même ensemble solution.

2-) (P5r) est une relaxation de (P3r).

Preuve : (Krarup et Pruzan (1983))

Montrons d'abord 2-)

Si (P5r) est une relaxation de (P3r) alors l'ensemble solution de (P3r) est contenu dans (P5r).

Soit X'_{ji} et y'_j une solution réalisable de (P3r). Puisque seules les contraintes (P3.2) et (P5.2) diffèrent dans les deux problèmes, ce sont les seules qui puissent rendre les deux problèmes non-équivalents.

Puisque X'_{ji} et y'_j représente une solution de (P3r), alors

$$X'_{ji} \leq y'_j \qquad \text{pour tout } i \in C, j \in D \qquad \text{(P3r.2)}$$

$$\Longleftrightarrow \sum_{i \in D} X'_{ji} \leq \sum_{i \in D} y'_j \qquad \text{pour tout } j \in D$$

$$\Longleftrightarrow \sum_{i \in D} X'_{ji} \leq y'_j \sum_{i \in D} 1 = y'_j n \qquad \text{pour tout } j \in D \qquad \text{(P5r.2)}$$

car n est le nombre de clients.

Donc une solution réalisable de (P3r) est également une solution réalisable de (P5r).

Preuve de 1-)

Il faut démontrer qu'une solution réalisable de (P5r), n'est pas nécessairement réalisable pour (P3r). Soit X'_{ji} et y'_j une solution réalisable de (P5r). Elle satisfait donc

$$\sum_{i \in C} X'_{ji} \leq n y'_j \qquad \text{pour tout } j \in D \qquad \text{(P5r.2)}$$

En particulier une telle solution peut avoir une de ces contraintes satisfaite avec égalité. Alors

$$\sum_{i \in C} X'_{ji} = n y'_j \qquad \text{pour tout } j \in D \qquad \text{(P5r.2)}$$

$$\Longleftrightarrow \sum_{i \in C} X'_{ji}/n = y'_j \qquad \text{pour tout } j \in D$$

Donc y'_j est en fait la moyenne des X'_{ji} ($i \in C$). Ainsi, en particulier, si les X'_{ji} ne sont pas tous égaux pour un j donné, alors un des X'_{ji} sera supérieur à leur moyenne ,y'_j. Il existe donc un X'_{ji} tel que

$$X'_{ji} > y'_j$$

Ainsi une contrainte de type (P3.2) n'est pas satisfaite. Donc cette solution n'est pas réalisable pour le problème (P3).

Donc les deux problèmes ne possèdent pas nécessairement le même ensemble de solutions réalisables.

c.q.f.d.

Dans cette section, nous avons démontré que les deux formulations possèdent les mêmes solutions entières, mais qu'une fois les contraintes d'intégralités retirées, les problèmes ne sont plus équivalents. Pour la suite du texte le problème (P3r) sera appelé la relaxation forte et (P5r) la relaxation faible. Ces deux problèmes peuvent être utilisés pour calculer une borne inférieure sur le problème original à chaque noeud du "branch-and-bound".

Mais puisque (P5r) est une relaxation de (P3r), la valeur de la fonction économique de (P5r) va être plus éloignée du problème original que celle de (P3r). Ceci entraîne un plus grand nombre de noeuds à explorer en utilisant (P5r).

Par contre, comme nous le verrons dans la section suivante, il existe une méthode directe pour résoudre (P5r). Une telle méthode n'existe pas pour (P3r).

Nous devons donc choisir entre la relaxation faible (P5r) et la relaxation forte (P3r).

La relaxation faible calcule donc une moins bonne borne inférieure à chaque noeud du "branch-and-bound". Mais puisqu'il est facile de le résoudre, peu de temps est nécessaire à chaque noeud. La relaxation forte va demander moins de noeuds, mais un temps de résolution plus grand à chaque noeud.

D'après les études faites jusqu'à présent(voir Krarup et Pruzan (1983) pour différentes méthodes de résolution avec les deux relaxations), la relaxation faible donne de meilleurs résultats. Nous allons donc présenter cette relaxation dans la section suivante.

2.3 – La relaxation faible

La formulation de la relaxation faible est la suivante:

$$\text{(P5r)} \quad \text{Min } Z = \sum_{j \in D} \sum_{i \in C} C_{ji} X_{ji} + \sum_{j \in D} f_j y_j$$

$$\text{sujet à} \quad \sum_{j \in D} X_{ji} = 1 \qquad \text{pour tout } i \in C \qquad \text{(P5r.1)}$$

$$\sum_{i \in C} X_{ji} \leq n y_j \qquad \text{pour tout } j \in D \qquad \text{(P5r.2)}$$

$$X_{ji} \geq 0 \qquad \text{pour tout } j \in D, i \in C \qquad \text{(P5r.3)}$$

$$y_j \geq 0 \qquad \text{pour tout } j \in D \qquad \text{(P5r.4)}$$

Pour la suite de cette section plaçons nous dans le contexte d'un noeud du "branch-and-bound". A ce noeud certaines variables sont fixées à 1, d'autres à 0 et celles restants sont libres. Définissons

K_0: ensemble des dépôts fixés fermés. $= \{j | y_j = 0\}$

K_1: ensemble des dépôts fixés ouverts. $= \{j | y_j = 1\}$

K_2: ensemble des autres dépôts. $= \{j | 0 < y_j < 1\}$

Puisque nous avons déjà démontré que (P5) est un cas particulier du modèle de Khumawala (P4), reprenons cette notation. Relaxons maintenant les contraintes forçant les y_j à être binaires pour obtenir (P4r) correspondant à la relaxation faible.

$$\text{(P4r)} \quad \text{Min } Z = \sum_{j \in D} \sum_{i \in P_j} C_{ji} X_{ji} + \sum_{j \in D} f_j y_j$$

$$\text{sujet à} \quad \sum_{j \in N_i} X_{ji} = 1 \qquad \text{pour tout } i \in C \qquad \text{(P4r.1)}$$

$$\sum_{i \in P_j} X_{ji} \leq n_j y_j \qquad \text{pour tout } j \in D \qquad \text{(P4r.2)}$$

$$X_{ji} \geq 0 \qquad \text{pour tout } j \in D, i \in C \qquad \text{(P4r.3)}$$

$$y_j \geq 0 \qquad \text{pour tout } j \in D \qquad \text{(P4r.4)}$$

Définissons également le modèle (P4rn), problème de relaxation faible, dans lequel les variables déjà fixées ont été remplacées par leurs valeurs. Cette notation représente donc le problème à résoudre pour obtenir une borne inférieure à un noeud quelconque du "branch-and-bound":

$$\text{(P4rn)} \quad Z = \sum_{j \in K_1} f_j + \min \left\{ \sum_{j \in (K_1 \cup K_2)} \sum_{i \in P_j} C_{ji} X_{ji} + \sum_{j \in K_2} f_j y_j \right\}$$

$$\text{sujet à} \sum_{j \in N_i} X_{ji} = 1 \qquad \text{pour tout } i \in C \qquad \text{(P4rn.1)}$$

$$\sum_{i \in P_j} X_{ji} \leq n_j \qquad \text{pour tout } j \in K_1 \qquad \text{(P4rn.2)}$$

$$\sum_{i \in P_j} X_{ji} \leq n_j y_j \qquad \text{pour tout } j \in K_2 \qquad \text{(P4rn.3)}$$

$$X_{ji} = 0 \qquad \text{pour tout } j \in K_0,\ i \in C$$

$$X_{ji} \geq 0 \qquad \text{pour tout } j \in (K_1 \cup K_2),\ i \in C \qquad \text{(P4rn.4)}$$

$$y_j \geq 0 \qquad \text{pour tout } j \in K_2 \qquad \text{(P4rn.5)}$$

Nous pouvons maintenant démontrer que ce problème possède la propriété d'affectation unique, comme le problème général de localisation simple. Cependant, nous allons d'abord démontrer deux résultats, qui nous permettrons de simplifier le modèle (P4rn).

Proposition 2.4 : (Khumawala (1972)) Dans le problème (P4rn), les contraintes de type (P4rn.2) sont redondantes.

Preuve : (Khumawala (1972)) Puisque par les contraintes (P4rn.4), les X_{ji} sont non-négatifs, et que pour chaque client la somme des X_{ji} y entrant est 1 (contraintes (P4rn.1)), tous les X_{ji} seront inférieurs ou égaux à 1.

Donc une contrainte de type (P4rn.2) est telle que

$$\sum_{i \in P_j} X_{ji} \leq \sum_{i \in P_j} 1 = n_j$$

puisque n_j représente le nombre d'éléments dans P_j.

Donc ces contraintes seront forcément respectées, étant donné les contraintes (P4rn.1) et (P4rn.4).

c.q.f.d.

Proposition 2.5 : (Khumawala (1972)) Dans une solution optimale de (P4rn), les contraintes de type (P4rn.3) tiennent avec l'égalité.

Preuve : (Khumawala (1972))

Supposons que les X'_{ji} et y'_j représentent une solution optimale de (P4rn). Pour un $j \in K_2$, nous avons alors

$$\sum_{i \in P_j} X'_{ji} < n_j y'_j \quad \text{, alors}$$
$$y'_j > \sum_{i \in P_j} X'_{ji}/n_j = Y \ .$$

Puisque cette contrainte est la seule dans laquelle y'_j apparaît, si la valeur de ce y'_j était remplacée par Y, alors le problème serait toujours réalisable. De plus la valeur de la fonction économique serait abaissée, puisque les f_j sont non- négatifs.

Ceci contredit l'hypothèse de départ d'une solution optimale.

c.q.f.d.

Puisque ces contraintes sont satisfaites avec égalité dans toute solution optimale, nous aurons

$$\sum_{i \in P_j} X_{ji} = n_j y_j \qquad \text{pour tout } j \in K_2$$
$$\Longleftrightarrow \ y_j = \sum_{i \in P_j} X_{ji}/n_j \qquad \text{pour tout } j \in K_2 \ .$$

En remplaçant ces y_j, par leurs valeurs dans (P4rn), nous obtenons le nouveau problème (P4rn1).

$$\text{(P4rn1)} \quad Z = \sum_{j \in K_1} f_j + \min \left\{ \sum_{j \in K_1} \sum_{i \in P_j} C_{ji} X_{ji} + \sum_{j \in K_2} \sum_{i \in P_j} [C_{ji} + (f_j/n_j)] X_{ji} \right\}$$

$$\text{sujet à} \sum_{j \in N_i} X_{ji} = 1 \qquad \text{pour tout } i \in C \qquad \text{(P4rn1.1)}$$

$$X_{ji} \geq 0 \qquad \text{pour tout } j \in (K_1 \cup K_2),\ i \in P_j \qquad \text{(P4rn1.2)}$$

A partir de ce modèle, nous pouvons maintenant démontrer que le problème de la relaxation faible possède la propriété d'affectation unique.

Proposition 2.6 : (Effroymson et Ray (1966)) A un noeud donné du "branch-and-bound", avec les ensembles de dépôts K_0, K_1 et K_2 connus, la solution optimale est:

$$X_{ji} = \begin{cases} 1 & \text{si } C_{ji} + g_j/n_j = \min\limits_{k \in K_1 \cup K_2} \{C_{ki} + g_k/n_k\} \\ 0 & \text{autrement} \end{cases}$$

et

$$y_j = \begin{cases} 0 & \text{si } j \in K_0 \\ 1 & \text{si } j \in K_1 \\ \{\sum\limits_{i \in C} X_{ji}/n_j\} & \text{si } j \in K_2. \end{cases}$$

où

$$g_k = \begin{cases} f_k & \text{si } k \in K_2 \\ 0 & \text{si } k \in K_1 \end{cases}$$

Preuve : (Effroymson et Ray (1966)).

1-) Pour les y_j:

Evident pour les dépôts j fixés et, par la proposition 2.5, trivial pour les j libres.

2-) Pour les X_{ji}:

Par les contraintes (P4rn1.1), le somme de tous les X_{ji} entrant à un client donné, est 1.

Mais, puisque chacun de ces X_{ji} possède un coefficient distinct dans la fonction économique, l'optimum sera alors obtenu lorsque le X_{ji} possédant le coefficient minimal est utilisé. Tous les autres X_{ji} de ce client seront fixés à 0.

c.q.f.d.

Ce résultat permet donc de trouver facilement une solution optimale de (P4rn1) à tout noeud de l'arbre du "branch-and-bound". Ce résultat est important parce que le compromis présenté en début de section entre le temps passé à chaque noeud et le nombre de noeuds à explorer dans l'arbre prend une autre dimension. Puisqu'une méthode directe existe pour calculer une solution optimale de (P4rn1) à chaque noeud, cette relaxation semble plus avantageuse que la relaxation forte.

En terminant, nous démontrons qu'une solution optimale de (P4rn1) est une solution réalisable du problème original (P4). Ceci est important puisque, lorsque la solution obtenue est entière dans l'algorithme du "branch-and-bound", la solution est réalisable pour le problème original et représente la meilleure solution possible étant donné la combinaison des y_j.

Proposition 2.7 : (Khumawala (1972)) Si à un noeud donné du "branch-and-bound" la solution optimale du problème (P4rn1) est entière, alors

1-) Cette solution est réalisable pour le problème original (P4);

2-) Cette solution est optimale dans (P4) pour la combinaison des y_j de cette solution.

Preuve : (Khumawala (1972))

preuve de 1-):

Pour prouver ce résultat, il suffit de démontrer que toutes les contraintes de (P4) sont satisfaites.

Les contraintes de type (P4.1) sont satisfaites puisqu'elles sont identiques aux contraintes de type (P4rn1.1).

Les contraintes de type (P4.2) sont satisfaites pour les dépôts fixés fermés, puisque les X_{ji} sortant de ces dépôts, sont éliminés. Pour les dépôts fixés ouverts, nous avons déjà démontré dans la proposition 2.4, que les contraintes sont redondantes.

Pour les dépôts libres, nous savons que

$$y_j = \sum_{i \in C} X_{ji}/n_j$$

$$\sum_{i \in C} X_{ji} = n_j y_j$$

ce qui correspond à une contrainte de type (P4.2) pour un dépôt libre.

Les contraintes de type (P4.3) correspondent aux contraintes de type (P4rn1.2) pour les dépôts libres et ouverts. Puisque les X_{ji} sont éliminés du problème pour les dépôts j fermés, alors ils sont nuls et satisfont également ces contraintes.

Les contraintes de type (P4.4) sont également satisfaites, puisque dans une solution entière les y_j sont 0 ou 1.

Donc toutes les contraintes de (P4) sont satisfaites, et ainsi la solution est réalisable pour ce problème.

Preuve de 2-):

Reprenons la fonction économique de (P4rn1)

$$Z = \sum_{j \in K_1} f_j + \min \left\{ \sum_{j \in K_1} \sum_{i \in P_j} C_{ji} X_{ji} + \sum_{j \in K_2} \sum_{i \in P_j} [C_{ji} + (f_j / n_j)] X_{ji} \right\}$$

Puisque la solution est entiere, nous avons deux possibilités pour les dépôts j libres:

Par la proposition 2.6, lorsqu'un dépôt j est libre

$$y_j = \sum_{i \in C} X_{ji} / n_j$$

Supposons que dans la solution optimale $y_j = 0$, alors

$$X_{ji} = 0 \qquad \text{pour tout } i \in P_j$$

Par contre si $y_j = 1$, alors

$$X_{ji} = 1 \qquad \text{pour tout } i \in P_j$$

Ceci puisque par la propriété d'affectation unique, les X_{ji} sont 0 ou 1. De plus n_j représente le nombre d'éléments dans P_j. Donc y_j ne peut valoir 1 que si tous les éléments de P_j sont 1.

Divisons l'ensemble K_2 en deux sous-ensembles. K_{2_0} contient les dépôts libres valant 0 dans la solution optimale. K_{2_1} contient les dépôts libres valant 1 dans la solution optimale.

En utilisant ces ensembles et en remplaçant les X_{ji} par leurs valeurs, la fonction économique devient

$$\sum_{j \in K_1} f_j + \min \left\{ \sum_{j \in K_1} \sum_{i \in P_j} C_{ji} X_{ji} + \sum_{j \in K_{2_1}} \sum_{i \in P_j} C_{ji} + \sum_{j \in K_{2_1}} \sum_{i \in P_j} f_j / n_j \right\}$$

qui devient

$$\sum_{j \in (K_1 \cup K_{2_1})} f_j + \min \left\{ \sum_{j \in (K_1 \cup K_{2_1})} \sum_{i \in P_j} C_{ji} X_{ji} \right\}$$

Ceci correspond à la fonction économique de (P4) dans lequel les arcs sortant de dépôts fermés seraient éliminés.

c.q.f.d.

2.4 – Méthode de Khumawala

Dans cette section, nous allons étudier la méthode de Khumawala (1972) qui utilise la relaxation faible pour résoudre le problème de localisation simple.

Pour pouvoir utiliser le "branch-and-bound", les éléments nécessaires doivent

être réunis. La proposition 2.6 permet de calculer une borne inférieure à chaque noeud. Une borne supérieure initiale peut être calculée par les méthodes proposées dans le chapitre précédent. Il reste à définir des méthodes de sélection de variables à fixer. Nous verrons qu'il est également possible d'effectuer des simplifications à certains noeuds du "branch-and-bound". Nous terminons par une présentation des résultats obtenus par Khumawala.

2.4.1 – Simplifications

Khumawala propose trois critères de simplification pour réduire la taille du problème et le nombre de noeuds à explorer. Deux de ces critères permettent dans certaines conditions de fixer un dépôt ouvert ou fermé. Le troisième critère permet, toujours sous certaines conditions, de réduire le nombre de clients pouvant être servis par un dépôt.

A chaque noeud du "branch-and-bound", il faut répéter l'application de ces trois critères jusqu'à ce qu'aucune autre amélioration ne soit possible. Ceci parce que l'application d'une simplification, peut permettre l'application d'une autre de ces simplifications.

2.4.1.1 – Simplification 1

Cette simplification détermine une borne inférieure sur la diminution des coûts engendrée par l'ouverture d'un dépôt. Si cette borne est positive alors il est avantageux d'ouvrir le dépôt. Mathématiquement

Pour $j \in K_2$ et $i \in P_j$

$$\nabla_{ji} = \min_{k \in N_i \cap (K_1 \cup K_2); k \neq j} \{\max\{C_{ki} - C_{ji}, 0\}\}$$

$$\Delta_j = \sum_{i \in P_j} \nabla_{ji} - f_j.$$

Si $\Delta_j > 0$ alors $y_j = 1$ (c'est-à-dire que le dépôt j est fixé ouvert) pour le sous-arbre associé à ce noeud.

Le ∇_{ji} est en fait une borne inférieure sur le coût de transport supplémentaire engendré par la fermeture du dépôt j, pour le client i. La somme pour tous les clients de ces ∇_{ji}, indique une borne inférieure sur les coûts de transport supplémentaires totaux occasionnés par la fermeture du dépôt j. Si ces coûts supplémentaires sont supérieurs au coût d'ouverture de j, alors le dépôt j doit être fixé ouvert pour l'exploration du sous-arbre résultant. Ce calcul est valable étant donné la propriété d'affectation unique de la relaxation faible.

2.4.1.2 – Simplification 2

Cette méthode donne un moyen de réduire la taille du problème pour tout le sous-arbre émanant du noeud où on effectue la simplification. En fait, cette simplification cherche à diminuer les n_j. Ceux-ci représentent les cardinalités des ensembles P_j des clients qui peuvent être servis par le candidat-dépôt j. Mathématiquement

si pour $j \in K_2$ et $i \in P_j$

$$\min_{k \in K_1 \cap N_i} \{C_{ki} - C_{ji}\} < 0$$

alors n_j est réduit de 1, le client i est retiré de P_j, et le dépôt j est retiré de N_i.

Puisque nous savons que dans une solution optimale du problème de localisation

simple, un client peut être servi seulement par le dépôt ouvert le plus rapproché, alors à un noeud donné de l'arbre du "branch-and-bound", si un client peut être servi à meilleur coût par un dépôt déjà ouvert, plutôt que par un dépôt libre, alors l'arc allant de ce dépôt libre vers le client ne sera jamais utilisé dans le sous-arbre résultant.

2.4.1.3 – Simplification 3

Cette dernière simplification détermine une borne maximale sur la réduction des coûts de transport entrainée par l'ouverture d'un dépôt. Si cette borne est négative alors il est avantageux de fermer le dépôt. Mathématiquement:

Pour $j \in K_2$ et $i \in P_j$

$$\omega_{ji} = \min_{k \in N_i \cap K_1} \{\max\{C_{ki} - C_{ji}, 0\}\}$$

$$\Omega_j = \sum_{i \in P_j} \omega_{ji} - f_j.$$

Si $\Omega_j < 0$ alors $y_j = 0$ (c'est-à-dire que le dépôt j est fixé fermé) pour le sous-arbre associé à ce noeud.

Le ω_{ji} est une borne supérieure sur le coût de transport supplémentaire engendré par la fermeture du dépôt j pour le client i. La somme pour tous les clients de ces ω_{ji} indique une borne supérieure sur les coûts de transport supplémentaires totaux occasionnés par la fermeture du dépôt j. Si ces coûts supplémentaires sont inférieurs au coût d'ouverture de j, alors le dépôt j doit être fixé fermé pour l'exploration du sous-arbre résultant. Ce calcul est valable étant donné la propriété

d'affectation unique de la relaxation faible.

2.4.2 – Méthodes de sélection

Une méthode de sélection est un processus consistant à choisir la variable à fixer. Dans un contexte de "branch-and-bound", à un noeud donné nous sélectionnons une variable libre dans K_2. Khumawala propose et analyse huit de ces méthodes(appelées aussi règles), mais d'autres peuvent être envisagées.

Rappelons que ces méthodes sont en général intuitives. Elles sont également appelées méthodes de branchement.

2.4.2.1 – Règles Delta

Ces règles utilisent les Δ_j calculés lors des simplifications. Dans ces simplifications si $\Delta_j > 0$ alors le dépôt j est fixé ouvert pour le sous-arbre découlant de ce noeud. Intuitivement, pour les Δ_j négatifs, ceux qui sont près de 0 (c'est-à-dire les plus élevés) ont une meilleure probabilité d'être ouverts dans la meilleure solution entière du sous-arbre résultant. Par contre pour les dépôts ayant les plus petits Δ_j, la probabilité est plus grande d'être fermés à une telle solution. Ces constatations nous mènent aux deux premières règles de sélection.

1)La règle du plus grand Delta.

Le dépôt libre ayant le plus grand Δ_j est choisi.

2)La régle du plus petit Delta.

Le dépôt libre ayant le plus petit Δ_j est choisi.

2.4.2.2 – Règles Oméga

Ces règles utilisent les Ω_j calculés pour chaque dépôt libre dans les simplifica-

tions. Comme il était mentionné alors, si $\Omega_j < 0$, le dépôt j est fixé fermé pour le sous-arbre résultant de ce noeud. Intuitivement, parmi les dépôts j dont le Ω_j est positif, ceux ayant un grand Ω_j ont une plus grande probabilité d'être ouverts dans la meilleure solution entière du sous-arbre résultant. De même ceux ayant de petits Ω_j, ont de plus grandes probabilités d'être fermés dans une telle solution. Deux règles découlent de ceci:

1)La règle du plus grand Oméga.

Le dépôt libre ayant le plus grand Ω_j est choisi.

2)La règle du plus petit Oméga.

Le dépôt libre ayant le plus petit Ω_j est choisi.

2.4.2.3 – Règles Y

Ici, il s'agit d'utiliser les valeurs obtenues pour les y_j lors du calcul de la solution optimale du noeud. Nous savons que les y_j libres sont compris entre 0 et 1. Intuitivement, si un y_j est près de 1, ce dépôt a une plus grande probabilité d'être ouvert, dans la meilleure solution entière du sous-arbre résultant. De même si le y_j est près de 0, le dépôt a une plus grande probabilité d'être fermé dans la meilleure solution entière du sous-arbre résultant. De ceci, les deux règles suivantes peuvent être tirées:

1)la règle du plus grand Y.

Le dépôt libre qui possède le y_j le plus près de 1 est choisi.

2)la règle du plus petit Y.

Le dépôt libre qui possède le y_j le plus près de 0 est choisi.

Une combinaison de ces deux règles pourrait également être intéressante. Il

s'agirait alors de choisir la variable la plus près de 0 ou la plus près de 1. Par exemple un y_j de 0.9 serait choisi avant un y_j de 0.2. Khumawala n'a cependant pas utilisé cette règle.

2.4.2.4 – Règles de la demande

Dans une solution de la relaxation faible à un noeud donné du "branch-and-bound", la demande de chaque client est satisfaite par un seul dépôt. Parmi les dépôts libres, ceux qui doivent satisfaire une grande demande, c'est-à-dire la somme des demandes des clients qu'ils desservent, ont une plus grande probabilité d'être ouverts dans la solution optimale. Par contre, ceux qui ont une faible demande à satisfaire, ont une faible probabilité d'être ouverts. Il s'agit donc de calculer pour chaque dépôt libre la somme des demandes des clients qu'il dessert. Deux règles peuvent être tirées de ce fait:

1)la règle de la plus grande demande.

Le dépôt libre possédant la plus grande demande est choisi.

2)la règle de la plus petite demande.

Le dépôt libre possédant la plus petite demande est choisi.

Notons que ces règles ne mentionnent pas s'il est préférable d'explorer en premier, le sous-arbre où la variable est fixée à 0, ou celui où elle est fixée à 1. Bien sûr, les deux sous-arbres doivent être explorés de toute façon. Cependant si celui qui possède la meilleure solution entière est exploré en premier, la borne supérieure peut être améliorée plus rapidement. Ceci peut entraîner une diminution du nombre de noeuds à explorer dans le deuxième sous-arbre.

D'autres règles de sélection pourraient aussi être imaginées et des combinaisons des règles mentionnées plus haut pourraient également être envisagées.

2.4.3 – Les résultats

Dans cette section nous présentons brièvement les résultats obtenus par Khumawala. Les tests portent sur les méthodes de sélection de variables.

Khumawala propose cinq critères de comparaison pour tester l'efficacité des méthodes de sélection mentionnées dans la section précédente. Ses critères sont:

1) Le nombre total d'itérations (noeuds). Ceci correspond à la taille de l'arbre du "branch-and-bound".
2) Le nombre d'itérations effectuées avant d'atteindre la solution optimale. Ceci reflète la vitesse à laquelle la solution optimale est atteinte.
3) Le nombre de noeuds distincts qui indiquent l'espace mémoire requis pour résoudre le problème. Ceci est en fait la profondeur atteinte dans l'arbre, c'est-à-dire le nombre maximal de variables qui ont été fixées avant d'atteindre une solution entière.
4) Le temps requis pour résoudre le problème.
5) La "qualité" de la solution atteinte, si l'arbre n'a pu être exploré au complet. Ceci arrive lorsqu'il y a trop de noeuds à explorer. La "qualité" est mesurée par l'éloignement par rapport à l'optimum.

Pour tester ces problèmes Khumawala a utilisé 16 problèmes tests contenant 25 candidats-dépôts et 50 clients. Seulement trois des huit règles proposées ont trouvé

la solution optimale pour tous les problèmes tests. Ces trois méthodes sont celles du plus grand Delta, du plus grand Oméga et du plus grand Y. Ces trois méthodes ont, dans tous les cas, exploré complètement l'arbre du "branch-and-bound", ce qui nous permet d'être certain que les solutions obtenues sont bien optimales.

Ces trois méthodes possèdent donc l'avantage de donner la solution optimale dans tous les problèmes tests. Cependant, elles ne le font pas toutes avec la même efficacité. Par exemple la règle du plus grand Delta a besoin en moyenne de 150 itérations par problème, tandis que les règles du plus grand Oméga et du Y, ont respectivement besoin de 45 et 95 itérations, en moyenne.

La règle du plus grand Oméga est également la meilleure en ce qui a trait aux autres critères. Pour atteindre la solution optimale, par exemple, elle n'a besoin que de 19 itérations en moyenne, comparativement à 59 et plus pour les autres méthodes. Elle n'a également besoin que de 11 noeuds distincts en moyenne, comparativement à 23 ou plus pour les autres. Finalement elle n'a besoin que de 3.8 secondes en moyenne, sur un ordinateur de type CDC 6500, pour résoudre un problème, contre 7.2 secondes et plus pour les autres méthodes.

2.5 – Conclusion

Dans ce chapitre, nous avons présenté et analysé le problème de localisation simple. Nous avons également présenté une méthode de résolution possible, utilisant la technique du "branch-and-bound".

Khumawala a obtenu de bons résultats avec cette méthode, c'est pourquoi nous utiliserons une méthode semblable pour solutionner le problème de localisation avec

équilibrage. Nous nous inspirons aussi de ses travaux pour dériver des règles et des critères de sélection de variables à fixer.

CHAPITRE 3 – LA LOCALISATION AVEC EQUILIBRAGE

Ce chapitre est consacré à l'étude du problème de localisation avec équilibrage. La définition de ce problème est présentée dans le chapitre sur la modélisation, nous passons donc directement à la présentation de la formulation mathématique du problème.

Dans ce chapitre, nous utiliserons, presque sans modification, la notation de Crainic, Dejax et Delorme (1986).

3.1 – Formulation

Dans cette section, nous reprenons la formulation du problème de localisation avec équilibrage présentée dans le chapitre sur la modélisation. Nous analysons plus en détail cette formulation. Nous présentons également certains cas particuliers permettant de réduire la taille du problème.

Le problème de localisation avec équilibrage peut être formulé ainsi (Crainic, Dejax, Delorme (1986)):

$$\text{(P1)} \qquad \text{Min } Z = \sum_{j \in D} f_j y_j + \sum_{p \in P} \left\{ \sum_{i \in C} \sum_{j \in D} \left(c_{ijp} x_{ijp} + c_{jip} x_{jip} \right) + \sum_{j \in D} \sum_{k \in D} s_{jkp} w_{jkp} \right\}$$

$$\text{sujet à } \sum_{j \in D} x_{ijp} = O_{ip}$$

$$\sum_{j \in D} x_{jip} = D_{ip} \qquad \text{pour tout } i \in C, p \in P \qquad \text{(P1.1)}$$

$$x_{ijp} \leq My_j$$

$$x_{jip} \leq My_j \qquad \text{pour tout } i \in C, j \in D, p \in P \qquad \text{(P1.2)}$$

$$w_{jkp} \leq My_j$$

$$w_{kjp} \leq My_j \qquad \text{pour tout } j, k \in D, p \in P \qquad \text{(P1.3)}$$

$$\sum_{i \in C} x_{ijp} + \sum_{k \in D} w_{kjp} - \sum_{i \in C} x_{jip} - \sum_{k \in D} w_{jkp} = 0$$
$$\text{pour tout } j \in D, p \in P \qquad \text{(P1.4)}$$

$$x_{ijp} \geq 0$$

$$x_{jip} \geq 0 \qquad \text{pour tout } i \in C, j \in D, p \in P$$

$$w_{jkp} \geq 0 \qquad \text{pour tout } j, k \in D, p \in P$$

$$y_j \in \{0,1\} \qquad \text{pour tout } j \in D \qquad \text{(P1.5)}$$

La taille de ce problème devient rapidement énorme. Ceci est surtout dû aux contraintes de type (P1.2) et (P1.3). En effet, une contrainte (P1.2) existe pour chaque variable de type x_{ijp} et x_{jip}. De même chaque variable de type w_{jkp} rend nécessaire la création de deux contraintes. Toutes ces contraintes servent à assurer que seuls les arcs entrant ou sortant de dépôts ouverts transportent du flot.

Cependant certaines simplifications sont possibles dans des cas particuliers. Par exemple les contraintes de type (P1.3) peuvent être éliminées.

Proposition 3.1 : (Crainic, Dejax, Delorme (1986)) Lorsque les fonctions de coûts interdépôts dans le modèle (P1) vérifient:

$$s_{ikp} \leq s_{ijp} + s_{jkp} \qquad \text{pour tout } i, j, k \in D \text{ et } p \in P,$$

alors les contraintes (P1.3)

$$w_{jkp} \leq M y_j$$

$$w_{kjp} \leq M y_j \qquad \text{pour tout } j, k \in D \text{ et } p \in P$$

sont redondantes.

Preuve : (Crainic, Dejax et Delorme (1986))

Appelons (P2) la formulation obtenue en enlevant les contraintes (P1.3) dans (P1).

Soit $u' = (y', x', w')$ une solution optimale de (P2).

Montrons que, sous les hypothèses énoncées, ou u' est aussi solution optimale de (P1), ou il est possible d'en déduire une solution u'' qui soit également optimale pour (P1).

Deux cas sont possibles.

1-) u' vérifie les contraintes (P1.3) de (P1). Alors u' est une solution optimale de (P1) car (P2) est une relaxation de (P1).

2-) u' ne vérifie pas les contraintes (P1.3).

Alors, il existe au moins $r \in P$, $l, h \in D$ tel que

$$w'_{lhr} \neq 0 \text{ et } y'_l = 0.$$

Mais (par les contraintes (P1.2))

$$y'_l = 0 \Rightarrow x'_{ilp} = x'_{lip} = 0 \qquad \text{pour tout } i \in C, p \in P$$

donc $\sum_{i \in C} x'_{ilp} = \sum_{i \in C} x'_{lip} = 0$ pour tout $p \in P$

donc (par les contraintes (P1.4))

$$\sum_{k \in D} w'_{klp} - \sum_{k \in D} w'_{lkp} = 0 \quad \text{pour tout } p \in P$$

et en particulier

$$\sum_{k \in D} w'_{klr} - \sum_{k \in D} w'_{lkr} = 0$$

avec

$$\sum_{k \in D} w'_{klr} \neq 0 \text{ et } \sum_{k \in D} w'_{lkr} \neq 0$$

puisque, par hypothèse,

$$w'_{lhr} \neq 0 \text{ et } w'_{klr} \geq 0,\ w'_{lkr} \geq 0 \quad \text{pour tout } k \in D.$$

Soit alors les ensembles

$$D_I = \{k | w'_{klr} \neq 0\} \text{ et } D_O = \{k | w'_{lkr} \neq 0\}$$

et construisons le réseau de transport biparti $G = (N, A)$ où:

- $N = D_I \cup D_O$
- $A = D_I \times D_O$
- demandes aux noeuds D_O = les flots w'_{lkr}
- offres aux noeuds D_I = les flots w'_{klr}
- coûts sur arcs $(k_1, k_2) \in A = s_{k_1 lr} + s_{lk_2 r}$.

Une solution optimale du problème de transport (notée v'_{kj}, $k \in D_I, j \in D_O$), associée au réseau précédent, permet de construire une nouvelle solution $u'' = (x'', y'', w'')$ de (P1) avec $x'' = x'$, $y'' = y'$ et w'' tel que

$$w''_{kjp} = \begin{cases} w'_{kjp} & \text{si } (k,j) \in D', \quad \text{pour tout } p \\ 0 & \text{si } p = r,\ j = l,\ k \in D_I \text{ ou si } p = r,\ k = l,\ j \in D_O \\ w'_{kjp} + v'_{kj} & \text{si } p = r,\ k \in D_I,\ j \in D_O \end{cases}$$

où $D' = D \times D \backslash \{(k,j) | k \in D_I \cup \{l\}, j \in D_O \cup \{l\}\}$.

Nous avons donc:

$$\sum_{p \in P} \sum_{k \in D} \sum_{j \in D} s_{kjp} w''_{kjp} = \sum_{p \in P} \sum_{(k,j) \in D'} s_{kjp} w'_{kjp} + \sum_{k \in D_I} \sum_{j \in D_O} s_{kjr} (w'_{kjp} + v'_{kj}) .$$

Or, par l'hypothèse de départ sur les coûts s_{kjp}:

$$\begin{aligned} \sum_{k \in D_I} \sum_{j \in D_O} s_{kjr} v'_{kj} &\leq \sum_{k \in D_I} \sum_{j \in D_O} (s_{klr} + s_{ljr}) v'_{kj} \\ &= \left(\sum_{k \in D_I} s_{klr} \sum_{j \in D_O} v'_{kj} \right) + \left(\sum_{j \in D_O} s_{ljr} \sum_{k \in D_I} v'_{kj} \right) \\ &= \sum_{k \in D} s_{klr} w'_{klr} + \sum_{j \in D} s_{ljr} w'_{ljr} \end{aligned}$$

D'où

$$\begin{aligned} \sum_{p \in P} \sum_{k \in D} \sum_{j \in D} s_{kjp} w''_{kjp} &\leq \sum_{p \in P} \sum_{k \in D} \sum_{j \in D} s_{kjp} w'_{kjp} \\ &\Rightarrow z(u'') \leq z(u') \end{aligned}$$

De plus, puisque u'' est réalisable pour (P1), que (P2) est une relaxation de (P1) et que u' est optimal pour (P2),

$$z(u'') = z(u')$$

c.q.f.d.

Pour la suite de ce chapitre, nous allons définir (P2) comme étant la formulation (P1) dans laquelle les contraintes de type (P1.3) sont retirées. Cette hypothèse est

réaliste, puisqu'en utilisant seulement le réseau formé des dépôts et des arcs les reliant, il est possible de calculer les chemins plus courts entre chaque paire de dépôts.

Un corollaire de ce fait est la proposition 3.2.

Proposition 3.2 : (Crainic, Dejax, Delorme (1986)) Lorsque

$$s_{ikp} \leq s_{ijp} + s_{jkp} \qquad \text{pour tout } i,j,k \in D \text{ et } p \in P,$$

une solution optimale au problème modifié (P2) est réalisable, donc optimale, pour le problème initial (P1).

La preuve de ceci se trouve dans la proposition 3.1.

Tentons maintenant d'évaluer la constante M pour le modèle (P2). Comme dans le problème de localisation simple, cette constante doit être suffisamment grande pour assurer que si un dépôt j est fixé ouvert aucune restriction sur le flot sortant ou entrant à ce dépôt n'existera.

Nous savons par les contraintes de type (P1.1) que chaque x_{ijp} est inférieur ou égal à l'offre du client i (O_{ip}) pour le produit p. Donc, pour les x_{ijp}, la constante M peut être remplacée par le O_{ip} correspondant. De la même manière, pour les x_{jip}, chacun est inférieur ou égal à son D_{ip} correspondant. Donc, pour les contraintes de type (P1.2) correspondant à ces variables, la constante M peut être remplacée par le D_{ip} correspondant.

Pour simplifier la présentation nous supposons dorénavant que l'inégalité stricte de la proposition 3.2 est vérifié. Nous remplaçons également la constante M par les valeurs de remplacements calculées. Le nouveau modèle obtenu est le suivant

(Crainic, Dejax et Delorme (1986)):

$$(L) \quad \text{Min } Z = \sum_{j \in D} f_j y_j + \sum_{p \in P} \left\{ \sum_{i \in C} \sum_{j \in D} (c_{ijp} x_{ijp} + c_{jip} x_{jip}) + \sum_{j \in D} \sum_{k \in D} s_{jkp} w_{jkp} \right\}$$

$$\text{sujet à } \sum_{j \in D} x_{ijp} = O_{ip}$$

$$\sum_{j \in D} x_{jip} = D_{ip} \qquad \text{pour tout } i \in C, p \in P \qquad (L.1)$$

$$x_{ijp} \leq O_{ip} y_j$$

$$x_{jip} \leq D_{ip} y_j \qquad \text{pour tout } i \in C, j \in D, p \in P \qquad (L.2)$$

$$\sum_{i \in C} x_{ijp} + \sum_{k \in D} w_{kjp} - \sum_{i \in C} x_{jip} - \sum_{k \in D} w_{jkp} = 0$$

$$\text{pour tout } j \in D, p \in P \qquad (L.4)$$

$$x_{ijp} \geq 0$$

$$x_{jip} \geq 0 \qquad \text{pour tout } i \in C, j \in D, p \in P$$

$$w_{jkp} \geq 0 \qquad \text{pour tout } j, k \in D, p \in P$$

$$y_j \in \{0,1\} \qquad \text{pour tout } j \in D \qquad (L.5)$$

Il reste à déterminer si le problème de localisation avec équilibrage possède la propriété d'affectation unique comme le problème de localisation simple. Ceci est possible seulement pour un cas particulier du problème.

Proposition 3.3 : (Crainic, Dejax, Delorme (1986)) Lorsque dans le modèle (L), l'inégalité du triangle est vérifiée pour l'ensemble des fonctions de coût, en particulier lorsque $c_{ijp} + s_{jkp} \geq c_{ikp}$ pour tout $i \in C$, $j,k \in D$, $p \in P$, il existe une solution optimale pour (L), où $w_{jkp} = 0$ pour tout $j,k \in D$, $p \in P$.

Preuve : (Crainic, Dejax, Delorme (1986))

Supposons que dans une solution optimale d'un problème satisfaisant les hypothèses, il existe un arc inter-dépôt (w_{jkp}) sur lequel le flot est non-nul.

Alors, ce flot vient d'un ou plusieurs clients. Appelons un de ces clients i. Le coût pour envoyer une unité du client i au dépôt k en passant par le dépôt j est $c_{ijp} + s_{jkp}$. Ce coût n'augmente pas en envoyant directement cette unité de i à k, par l'hypothèse de départ. Ceci ne change rien au reste du flot dans cette solution. Donc la valeur de la solution n'est pas augmentée, ce qui est impossible par hypothèse.

En répétant ce processus tant que des arcs inter-dépôts sont utilisés, nous obtenons une nouvelle solution optimale n'utilisant pas les arcs inter-dépôts.

c.q.f.d.

Dans ce cas le modèle peut s'écrire:

$$(LS) \quad \text{Min } Z = \sum_{j \in D} f_j y_j + \sum_{p \in P} \sum_{i \in C} \sum_{j \in D} (c_{ijp} x_{ijp} + c_{jip} x_{jip})$$

$$\text{sujet à} \quad \sum_{j \in D} x_{ijp} = O_{ip}$$

$$\sum_{j \in D} x_{jip} = D_{ip} \qquad \text{pour tout } i \in C,\ p \in P$$

$$x_{ijp} \leq O_{ip} y_j$$

$$x_{jip} \leq D_{ip} y_j \qquad \text{pour tout } i \in C,\ j \in D,\ p \in P$$

$$\sum_{i \in C} x_{ijp} - \sum_{i \in C} x_{jip} = 0 \qquad \text{pour tout } j \in D,\ p \in P$$

$$x_{ijp} \geq 0$$

$$x_{jip} \geq 0 \qquad \text{pour tout } i \in C,\ j \in D,\ p \in P$$

$$w_{jkp} \geq 0 \qquad \text{pour tout } j, k \in D,\ p \in P$$

$$y_j \in \{0, 1\} \qquad \text{pour tout } j \in D$$

Ce modèle ne présente pas un grand intérêt pratique, mais peut servir à démontrer la raison pour laquelle le problème de localisation avec équilibrage ne possède pas la propriété d'affectation unique. Dans le cas d'un modèle de localisation simple, toute la demande d'un client est satisfaite à partir du dépôt ouvert ayant le coût de transport le plus bas puisque chaque dépôt a une capacité infinie et peut fournir n'importe quelle quantité demandée.

Mais dans le modèle (LS), pour que la demande d'un client soit satisfaite par le dépôt le plus rapproché, ce dépôt doit avoir d'abord reçu suffisamment d'unités

venant des clients. Donc, pour tous les clients ayant le même dépôt le plus rapproché, la somme de leurs offres doit être égale à celle de leurs demandes, sinon l'offre d'un des clients devra être envoyée à un autre dépôt, ou la demande d'un des clients devra être satisfaite par un autre dépôt.

Présentons maintenant les relaxations faibles (WRL) et fortes (SRL) du problème de localisation avec équilibrage.

Nous définissons la relaxation forte du problème de localisation avec équilibrage comme étant le problème dans lequel les contraintes forçant les y_j à être des variables binaires sont éliminées. La relaxation forte s'énonce donc ainsi:

$$\text{(SRL)} \quad \text{Min } Z = \sum_{j \in D} f_j y_j + \sum_{p \in P} \left\{ \sum_{i \in C} \sum_{j \in D} (c_{ijp} x_{ijp} + c_{jip} x_{jip}) + \sum_{j \in D} \sum_{k \in D} s_{jkp} w_{jkp} \right\}$$

$$\text{sujet à} \quad \sum_{j \in D} x_{ijp} = O_{ip}$$

$$\sum_{j \in D} x_{jip} = D_{ip} \qquad \text{pour tout } i \in C,\ p \in P \qquad \text{(SRL.1)}$$

$$x_{ijp} \leq O_{ip} y_j$$

$$x_{jip} \leq D_{ip} y_j \qquad \text{pour tout } i \in C,\ j \in D,\ p \in P \qquad \text{(SRL.2)}$$

$$\sum_{i \in C} x_{ijp} + \sum_{k \in D} w_{kjp} - \sum_{i \in C} x_{jip} - \sum_{k \in D} w_{jkp} = 0$$

$$\text{pour tout } j \in D,\ p \in P \qquad \text{(SRL.4)}$$

$$x_{ijp} \geq 0$$

$$x_{jip} \geq 0 \qquad \text{pour tout } i \in C,\ j \in D,\ p \in P$$

$$w_{jkp} \geq 0 \qquad \text{pour tout } j, k \in D,\ p \in P$$

$$y_j \geq 0 \qquad \text{pour tout } j \in D \qquad \text{(SRL.5)}$$

La relaxation faible du problème de localisation avec équilibrage consiste à agréger les contraintes de type (L.2) en des contraintes de type (WLR.2). La formulation de ce nouveau problème est la suivante:

$$\text{(WRL)} \quad \text{Min } Z = \sum_{j \in D} f_j y_j + \sum_{p \in P} \left\{ \sum_{i \in C} \sum_{j \in D} (c_{ijp} x_{ijp} + c_{jip} x_{jip}) + \sum_{j \in D} \sum_{k \in D} e_{jkp} w_{jkp} \right\}$$

$$\text{sujet à} \quad \sum_{j \in D} x_{ijp} = O_{ip}$$

$$\sum_{j \in D} x_{jip} = D_{ip} \qquad \text{pour tout } i \in C, p \in P \qquad \text{(WRL.1)}$$

$$\sum_{p \in P} \sum_{i \in C} (x_{ijp} + x_{jip}) \leq \left\{ \sum_{p \in P} \sum_{i \in C} (O_{ip} + D_{ip}) \right\} y_j$$

$$\text{pour tout } j \in D \qquad \text{(WRL.2)}$$

$$\sum_{i \in C} x_{ijp} + \sum_{k \in D} w_{kjp} - \sum_{i \in C} x_{jip} - \sum_{k \in D} w_{jkp} = 0$$

$$\text{pour tout } j \in D, p \in P \qquad \text{(WRL.4)}$$

$$x_{ijp} \geq 0$$

$$x_{jip} \geq 0 \qquad \text{pour tout } i \in C, j \in D, p \in P$$

$$w_{jkp} \geq 0 \qquad \text{pour tout } j, k \in D, p \in P$$

$$y_j \geq 0 \qquad \text{pour tout } j \in D \qquad \text{(WRL.5)}$$

Ces deux relaxations peuvent être utilisées pour le calcul d'une borne inférieure dans un algorithme de "branch-and-bound". L'avantage de la relaxation forte est

quelle est plus près du problème original et permet donc de calculer une meilleure borne inférieure.

Par contre, comme nous le verrons dans la section suivante, la structure de la relaxation faible est un problème de multiflot à coût minimum sans capacité qui est facile à résoudre. La borne sera donc moins bonne, mais le temps de calcul à chaque noeud sera réduit.

3.2 – Le "branch-and-bound"

Dans cette section nous présentons une méthode de résolution pour le problème de localisation avec équilibrage. Cette méthode est un algorithme de "branch-and-bound" dans lequel le modèle de la relaxation faible est utilisé pour le calcul d'une borne inférieure à chaque noeud. Dans cette section, nous présentons donc les éléments nécessaires à un algorithme de "branch-and-bound".

3.2.1 – Calcul d'une borne inférieure

Plaçons nous maintenant dans le contexte d'un noeud quelconque du "branch-and-bound". Un tel noeud est caractérisé par l'état des dépôts. Ces derniers sont divisés en trois ensembles distincts.

K_0: ensemble des dépôts fixés fermés. $= \{j | y_j = 0\}$

K_1: ensemble des dépôts fixés ouverts. $= \{j | y_j = 1\}$

K_2: ensemble des autres dépôts. $= \{j | 0 < y_j < 1\}$

En remplaçant les variables des dépôts fixées par leurs valeurs, dans le modèle

(WRL), nous obtenons le nouveau modèle (WRL1).

$$\text{(WRL1)}\quad Z = \sum_{j\in K_1} f_j + \min\left\{ \sum_{j\in K_2} f_j y_j + \right.$$

$$\sum_{p\in P}\left\{ \sum_{i\in C}\sum_{j\in(K_1\cup K_2)} (c_{ijp}x_{ijp} + c_{jip}x_{jip}) + \right.$$

$$\left.\left. \sum_{j\in(K_1\cup K_2)}\sum_{k\in(K_1\cup K_2)} s_{jkp}w_{jkp} \right\}\right\}$$

sujet à $\sum_{j\in D} x_{ijp} = O_{ip}$

$$\sum_{j\in D} x_{jip} = D_{ip} \quad \text{pour tout } i \in C,\ p \in P \qquad \text{(WRL1.1)}$$

$$\sum_{p\in P}\sum_{i\in C}(x_{ijp} + x_{jip}) \le \left\{ \sum_{p\in P}\sum_{i\in C}(O_{ip} + D_{ip}) \right\} y_j$$

$$\text{pour tout } j \in K_2 \qquad \text{(WRL1.2)}$$

$$\sum_{i\in C} x_{ijp} + \sum_{k\in D} w_{kjp} - \sum_{i\in C} x_{jip} - \sum_{k\in D} w_{jkp} = 0$$

$$\text{pour tout } j \in (K_1 \cup K_2),\ p \in P \qquad \text{(WRL1.4)}$$

$$x_{ijp} = 0$$

$$x_{jip} = 0 \quad \text{pour tout } i \in C,\ j \in K_0,\ p \in P$$

$$x_{ijp} \ge 0$$

$$x_{jip} \ge 0 \quad \text{pour tout } i \in C,\ j \in (K_1 \cup K_2),\ p \in P$$

$$
\begin{aligned}
w_{jkp} &= 0 \quad \text{pour tout } j \in K_0, k \in D, p \in P, \\
&\qquad \text{ou } j \in D, k \in K_0, p \in P \\
w_{jkp} &\geq 0 \quad \text{pour tout } j, k \in (K_1 \cup K_2), \\
&\qquad p \in P \\
y_j &\geq 0 \quad \text{pour tout } j \in K_2 \qquad \text{(WRL1.5)}
\end{aligned}
$$

Remarquons que dans ce problème, tous les arcs entrant ou sortant d'un dépôt fermé ont des valeurs nulles. Ceci est dû au fait qu'en remplaçant le y_j dans une contrainte de type (WRL.4) par 0, la somme des flots sur les arcs entrant (x_{ijp}) et sortant (x_{jip}) des clients à partir du dépôt j devra être inférieure ou égale à 0. Puisque toutes ces variables sont contraintes à être non-négatives, elles ne peuvent qu'être nulles dans ce cas. Donc, aucun arc venant des clients vers ce dépôt j n'amène de flot au dépôt. De plus, aucun arc n'envoie de flot vers les clients en partant de ce dépôt. Ainsi le dépôt j ne peut servir que de dépôt de transition pour du flot venant d'un autre dépôt et allant vers un troisième. Ceci est cependant impossible lorsque l'inégalité du triangle est respecté, pour les arcs inter-dépôts (proposition 3.2).

Démontrons maintenant le résultat suivant:

Proposition 3.4 : (Crainic, Dejax et Delorme (1986))

Dans une solution optimale du problème (WRL1), les contraintes (WRL1.2) tiennent avec l'égalité.

Preuve : (Crainic, Dejax et Delorme (1986))

Supposons que dans une solution optimale du problème une contrainte de type (WRL1.2) est telle que

$$\sum_{p\in P}\sum_{i\in C}(x_{ijp}+x_{jip}) < \left\{\sum_{p\in P}\sum_{i\in C}(O_{ip}+D_{ip})\right\}y_j \quad j\in K_2$$

alors $y_j > \sum_{p\in P}\sum_{i\in C}(x_{ijp}+x_{jip}) / \sum_{p\in P}\sum_{i\in C}(O_{ip}+D_{ip}) = Y$.
Remplaçons la valeur de ce y_j par la constante Y. Le problème demeure réalisable puisque cette contrainte est la seule dans laquelle ce y_j apparaît. De plus la valeur de la fonction économique va être abaissée, puisque les f_j sont non-négatifs. Ceci contredit l'hypothèse de départ voulant que la solution soit au départ optimale.

c.q.f.d.

Remplaçons donc dans (WRL1) les y_j associés aux dépôts libres par les valeurs calculées dans la proposition précédente. C'est-à-dire

$$y_j = \sum_{p\in P}\sum_{i\in C}(x_{ijp}+x_{jip})/\Delta \qquad \text{pour tout } j\in K_2$$

où $\Delta = \sum_{p\in P}\sum_{i\in C}(O_{ip}+D_{ip})$.

Un y_j libre représente la fraction de la somme des offres et des demandes, venant des clients vers j et retournant aux clients en provenance de j. En remplaçant ces y_j par leurs valeurs dans la formulation de la relaxation faible, nous obtenons

$$\text{(WRL2)} \quad Z = \sum_{j \in K_1} f_j + \min \left\{ \sum_{p \in P} \left\{ \sum_{i \in C} \sum_{j \in K_1} (c_{ijp} x_{ijp} + c_{jip} x_{jip}) + \right. \right.$$

$$\sum_{i \in C} \sum_{j \in K_2} ((c_{ijp} + f_j/\Delta) x_{ijp} + (c_{jip} + f_j/\Delta) x_{jip}) +$$

$$\left. \left. \sum_{j \in (K_1 \cup K_2)} \sum_{k \in (K_1 \cup K_2)} s_{jkp} w_{jkp} \right\} \right\}$$

sujet à

$$\sum_{j \in (K_1 \cup K_2)} x_{ijp} = O_{ip}$$

$$\sum_{j \in (K_1 \cup K_2)} x_{jip} = D_{ip} \quad \text{pour tout } i \in C,\ p \in P \qquad \text{(WRL2.1)}$$

$$\sum_{i \in C} x_{ijp} + \sum_{k \in (K_1 \cup K_2)} w_{kjp} - \sum_{i \in C} x_{jip} - \sum_{k \in (K_1 \cup K_2)} w_{jkp} = 0$$

$$\text{pour tout } j \in (K_1 \cup K_2),\ p \in P \qquad \text{(WRL2.4)}$$

$x_{ijp} \geq 0$

$x_{jip} \geq 0$ pour tout $i \in C,\ j \in (K_1 \cup K_2),\ p \in P$

$w_{jkp} = 0$ pour tout $j \in K_0,\ k \in D,\ p \in P,$

ou $j \in D,\ k \in K_0,\ p \in P$

$w_{jkp} \geq 0$ pour tout $j, k \in (K_1 \cup K_2),\ p \in P$

$y_j \geq 0$ pour tout $j \in K_2$ (WRL1.5)

Proposition 3.5 : (Crainic, Dejax et Delorme (1986))

A tout noeud du "branch-and-bound"

$$\sum_{j \in K_2} y_j \leq 1 .$$

Preuve : (Crainic, Dejax et Delorme (1986))

Posons pour tout $j \in D$

$$W_j = \sum_{i \in C} \sum_{p \in P} (x'_{ijp} + x'_{jip}) / \sum_{i \in C} \sum_{p \in P} (O_{ip} + D_{ip})$$

où x'_{ijp}, x'_{jip} et y'_j font partie de la solution optimale obtenue à un noeud quelconque du "branch-and-bound".

En particulier $y'_j = W_j$ pour tout $j \in K_2$.

Donc

$$\sum_{j \in K_2} y'_j \leq \sum_{j \in D} W_j .$$

$$\sum_{j \in D} W_j = \sum_{j \in D} \{ \sum_{i \in C} \sum_{p \in P} (x'_{ijp} + x'_{jip}) / \sum_{i \in C} \sum_{p \in P} (O_{ip} + D_{ip}) \}$$

$$= \sum_{j \in D} \sum_{i \in C} \sum_{p \in P} (x'_{ijp} + x'_{jip}) / \sum_{i \in C} \sum_{p \in P} (O_{ip} + D_{ip})$$

Puisque le numérateur est en fait la somme de tous les $x'ijp$ et des x'_{jip}, alors ce numérateur est en fait la somme de toutes les offres et de toutes les demandes.

Donc $\sum_{j \in D} W_j = 1$.

c.q.f.d.

Proposition 3.6 : (Crainic, Dejax, Delorme (1986)) (WRL2) est un problème de multiflot à coût minimum sans capacité.

Preuve : (Crainic, Dejax et Delorme (1986)).

Considérons le réseau $G = (N, A)$ utilisé pour la définition du problème de localisation avec équilibrage dans le premier chapitre.

Ce réseau est défini comme suit:

$N = C \cup D' \cup C'$ et $A = A_1 \cup A_2 \cup A_3$ où

C = ensemble des clients ayant une offre (sources);

$D' = K_1 \cup K_2$ = ensemble des dépôts non-fermés;

C' = ensemble des clients ayant une demande (puits);

et

$A_1 = \{(r, s) | r \in C, s \in D'\}$=ensemble des arcs clients-dépôts

$A_2 = \{(r, s) | r \in D', s \in C'\}$=ensemble des arcs dépôts-clients

$A_3 = \{(r, s) | r, s \in D'\}$=ensemble des arcs inter-dépôts.

Posons également

$c'_{ijp} = c_{ijp}$ et $c'_{jip} = c_{jip}$ pour tout $i \in C$, $j \in K_1$, $p \in P$

$c'_{ijp} = c_{ijp} + f_j/\Delta$ et $c'_{jip} = c_{jip} + f_j/\Delta$ pour tout $i \in C$, $j \in K_2$, $p \in P$

De plus, soit pour tout $p \in P$

$$\begin{cases} a_{ijp} = c'_{ijp} & \text{et } z_{ijp} = x_{ijp} \quad \text{pour tout } (i,j) \in A_1, \\ a_{ji'p} = c'_{jip} & \text{et } z_{ji'p} = x_{jip} \quad \text{pour tout } (j,i') \in A_2, \\ a_{jkp} = s_{jkp} & \text{et } z_{jkp} = w_{jkp} \quad \text{pour tout } (j,k) \in A_3. \end{cases}$$

Le problème (WRL2) peut alors être réécrit comme suit

$$Z = \min \sum_{p \in P} \sum_{(r,s) \in A} a_{rsp} z_{rsp}$$

sujet à

$$\sum_{(r,s)\in A} z_{rsp} - \sum_{(s,r)\in A} z_{srp} = \begin{cases} O_{rp} & \text{pour tout } r \in C,\ p \in P \\ 0 & \text{pour tout } r \in D',\ p \in P \\ -D_{rp} & \text{pour tout } r \in C',\ p \in P \end{cases}$$

$z_{rsp} \geq 0$ pour tout $(r,s) \in A,\ p \in P$

ce qui est un problème de multiflot à coût minimum, selon la définition présentée dans le premier chapitre.

c.q.f.d.

Ainsi (WRL2) est décomposable en l (nombre de produits) sous-problèmes de flot mutuellement indépendants. Ceci puisqu'aucune contrainte ne regroupe plusieurs produits.

Nous terminons cette section en démontrant que lorsqu'une solution entière est atteinte à un noeud de l'arbre du "branch-and-bound", cette solution est réalisable pour le problème original (L) et est également optimale pour le problème (L) dans lequel toutes les variables binaires seraient fixés aux valeurs de cette solution.

Proposition 3.7 : Si à un noeud quelconque du "branch-and-bound" la solution optimale du problème (WRL2) est entière, alors

1-) Cette solution est réalisable pour le problème original (L);

2-) Cette solution est optimale pour le problème (L) dans lequel les variables y_j déjà fixées sont remplacées par leurs valeurs.

Preuve : Preuve de 1-):

La solution obtenue sera réalisable pour (L) si toutes ses contraintes sont satisfaites.

D'abord les contraintes de type (WRL2.1) sont identiques aux contraintes (L.1).

Les contraintes de type (L.2) sont également satisfaites, puisque par la non-négativité des variables, les contraintes (L.1) assurent que les x_{ijp} et x_{jip} sont inférieurs ou égaux respectivement aux O_{ip} et D_{ip} correspondants. Donc lorsqu'un y_j vaut 1, les contraintes de type (L.2) sont satisfaites. Par contre si un y_j est nul, tous les arcs entrant ou sortant sont également nuls, donc les contraintes (L.2) sont satisfaites.

Les contraintes de type (L.4) sont équivalentes aux contraintes (WRL2.4) pour les dépôts libres et ouverts. Pour les dépôts fermés, puisque le flot sur les arcs entrant et sortant de ces dépôts sont nuls, les contraintes (L.4) sont également satisfaites dans ce cas.

Les autres contraintes assurent la non-négativité des variables, et sont donc satisfaites.

Puisque la solution obtenue satisfait toutes les contraintes du problème (L), alors la solution est réalisable pour ce problème.

Preuve de 2-):

Il s'agit de démontrer qu'en remplaçant les variables binaires par leurs valeurs dans la fonction économique de (L), nous obtiendrons la fonction économique de (WRL2).

Prenons la fonction économique de (WRL2).

Par la proposition 3.4, nous savons que lorsqu'un dépôt j est libre

$y_j = \sum_{p \in P} \sum_{i \in C} (x_{ijp} + x_{jip}) / \sum_{p \in P} \sum_{i \in C} (O_{ip} + D_{ip})$.

Mais, puisque la solution est entière les y_j sont entiers, même pour les j libres.

Ceci implique que si $y_j = 0$ alors tous les x_{ijp} et x_{jip} correspondants sont également nuls.

Par contre si $y_j = 1$, tout le flot du problème passe uniquement par le dépôt j. Dans le calcul de ce y_j le dénominateur est la somme totale des offres et des demandes. Puisque cette somme représente le flot total de ce qui est expédié et reçu par les clients, tout le flot du problème doit passer par ce dépôt. Donc tous les x_{ijp} entrant à j valent O_{ip} et tous les x_{jip} en sortant valent D_{ip}. Donc un seul dépôt libre peut posséder un $y_j = 1$.

Définissons deux sous-ensembles de K_2. K_{2_0}, ensemble des dépôts libres valant 0 dans la solution optimale. K_{2_1} ensemble des dépôts libres valant 1 dans la solution optimale.

Puisque les y_j ne font pas explicitement partie de la fonction économique de (WRL2), remplaçons les x_{ijp} et x_{jip} par leurs valeurs dans cette fonction. Nous obtenons alors

$$Z = \sum_{j \in K_1} f_j + \min \left\{ \sum_{p \in P} \left\{ \sum_{i \in C} \sum_{j \in K_1} (c_{ijp} x_{ijp} + c_{jip} x_{jip}) + \right. \right.$$

$$\sum_{i \in C} \sum_{j \in K_{2_1}} ((c_{ijp} + f_j/\Delta) O_{ip} + (c_{jip} + f_j/\Delta) D_{ip}) +$$

$$\left. \left. \sum_{j \in (K_1 \cup K_{2_1})} \sum_{k \in (K_1 \cup K_{2_1})} s_{jkp} w_{jkp} \right\} \right\} .$$

Prenons le terme

$$\sum_{p \in P} \sum_{i \in C} \sum_{j \in K_{2_1}} ((c_{ijp} + f_j/\Delta) O_{ip} + (c_{jip} + f_j/\Delta) D_{ip}) \quad ,$$

Cette expression peut être réarrangée pour obtenir

$$\sum_{p \in P} \sum_{i \in C} \sum_{j \in K_{2_1}} ((f_j/\Delta) O_{ip} + (f_j/\Delta) D_{ip}) + \sum_{p \in P} \sum_{i \in C} \sum_{j \in K_{2_1}} (c_{ijp} O_{ip} + c_{jip} D_{ip})$$

.

Cette expression est équivalente à

$$\sum_{j \in K_{2_1}} (f_j/\Delta) \sum_{p \in P} \sum_{i \in C} (O_{ip} + D_{ip}) + \sum_{p \in P} \sum_{i \in C} \sum_{j \in K_{2_1}} (c_{ijp} O_{ip} + c_{jip} D_{ip}) \quad .$$

Mais nous savons que

$$\Delta = \sum_{p \in P} \sum_{i \in C} (O_{ip} + D_{ip}) \quad ,$$

Donc l'expression précédante devient

$$\sum_{j \in K_{2_1}} f_j + \sum_{p \in P} \sum_{i \in C} \sum_{j \in K_{2_1}} (c_{ijp} O_{ip} + c_{jip} D_{ip}) \quad .$$

Ainsi la fonction économique devient

$$Z = \sum_{j \in (K_1 \cup K_{2_1})} f_j + \min \left\{ \sum_{p \in P} \left\{ \sum_{i \in C} \sum_{j \in K_1} (c_{ijp} x_{ijp} + c_{jip} x_{jip}) + \right. \right.$$

$$\sum_{i \in C} \sum_{j \in K_{2_1}} (c_{ijp} O_{ip} + c_{jip} D_{ip}) +$$

$$\left. \left. \sum_{j \in (K_1 \cup K_{2_1})} \sum_{k \in (K_1 \cup K_{2_1})} s_{jkp} w_{jkp} \right\} \right\} .$$

En remplaçant les O_{ip} et les D_{ip} par les variables x_{ijp} et x_{jip} correspondantes ($j \in K_{2_1}$), nous obtenons la fonction économique de (L) dans laquelle les variables binaires ont été fixées.

c.q.f.d.

3.2.2 – Calcul d'une borne supérieure

A chaque noeud de l'arbre du "branch-and-bound", il est possible de calculer une solution entière en fixant les dépôts libres ayant leur y_j non nul à 1. Cette solution est alors réalisable pour le problème original et si elle est inférieure à la borne supérieure, elle remplace cette dernière.

3.2.3 – Les simplifications

Dans cette section nous verrons pourquoi il est impossible de déterminer des simplifications analogues à celles utilisées dans l'algorithme proposé par Khumawala (1972). Cette impossibilité est dûe au fait que le problème de localisation avec équilibrage ne possède pas la propriété d'affectation unique de la localisation simple.

Ces simplifications sont basées sur le fait que l'ouverture d'un dépôt supplémentaire, dans une solution entière, ne peux que diminuer le flot sortant des dépôts déjà ouverts. De même, la fermeture d'un dépôt ouvert dans une solution entière, ne peut qu'augmenter le flot sortant des autres dépôts ouverts. Malheureusement, ces faits ne sont pas respectés pour le problème de localisation avec équilibrage, étant donné l'équilibrage possible entre les dépôts.

Supposons que nous avons une solution entière du problème de localisation

avec équilibrage. Dans cette solution nous fermons un des dépôts qui étaient fixés ouverts. Dans le problème de localisation simple, une telle intervention ne peut qu'augmenter le flot sur les autres dépôts ouverts. Cependant, pour ce problème, si le dépôt fermé reçoit ou envoie du flot à un autre dépôt, il est possible que ce flot, qui doit maintenant utiliser un nouveau chemin, soit obligé d'utiliser de nouveaux dépôts dans ce chemin, et ainsi la fermeture d'un dépôt, entrainerait la diminution du flot passant par un autre.

Par contre en prenant une solution entière de notre problème et en ouvrant un dépôt qui y était fermé, il est possible qu'un nouveau chemin utilisant ce nouveau dépôt et un autre dépôt déjà ouvert, soit utilisé. Cet autre dépôt verrait alors le flot y passant augmenter, malgré l'ouverture d'un dépôt supplémentaire.

Ces raisons expliquent pourquoi il est impossible de calculer aisément des simplifications à un noeud donné du "branch-and-bound" sur le probleme de localisation avec équilibrage.

3.2.4 – Méthodes de sélection de variables à fixer

Dans cette section nous présentons sept méthodes de sélection de variables à fixer pour le "branch-and-bound" pour le problème de localisation avec équilibrage. Contrairement au problème de localisation simple, les méthodes développées suggèrent également l'ordre dans lequel les deux noeuds générés suite au branchement sont traités.

Pour chacune de ces méthodes, nous supposons le contexte d'un noeud du "branch-and-bound" pour le modèle de relaxation faible dans lequel les y_j déjà

fixés ont été remplacés. Cette solution est représentées par les variables x'_{ijp}, x'_{jip}, w'_{kjp} et y'_j, pour tout $i \in C$, $j, k \in D$, $p \in P$.

3.2.4.1 – Règles Y

Ces règles sont semblables à celles du même nom proposées par Khumawala pour le problème de localisation simple. Comme nous avons déjà démontré qu'à chaque noeud du "branch-and-bound" chaque variable y_j contenue dans la solution permettant d'obtenir la borne inférieure est compris entre 0 et 1, alors cette méthode consiste à utiliser les y'_j les plus près de 1 ou de 0 comme critère de sélection. Intuitivement, cette méthode suppose qu'un y'_j près de 0, aura une plus grande probabilité d'être à 0 dans la meilleure solution entière contenue dans le sous-arbre résultant. De même, un y'_j près de 1, aura une plus grande probabilité d'être à 1 dans la meilleure solution entière contenue dans le sous-arbre résultant.

Nous n'utiliserons qu'une méthode de branchement pour cette règle, combinant les deux types de branchement possible. Cette méthode choisit comme variable à fixer, la variable étant la plus près de 0 ou de 1, et la fixe d'abord à cette valeur.

3.2.4.2 – Règles Y modifiées

Le principe de cette règle est de tenir compte de la valeur obtenue pour le y'_j, mais également du coût d'ouverture associé (f_j) au dépôt j. Intuitivement, cette règle utilise deux principes. D'abord celui de la règle précédente sur la valeur des y'_j. Le deuxième principe est qu'un dépôt ayant un coût d'ouverture élevé possède une plus grande probabilité d'être fermé dans la meilleure solution entière du sous-arbre résultant.

Un coefficient combinant ces deux principes est calculé pour chaque dépôt libre.

Ce coefficient est

$$Ymod_j = f_j(1 - y'_j)$$

1)la règle du plus grand Y modifié.

Le dépôt libre possédant le plus grand $Ymod_j$ est choisi et le dépôt j est tout d'abord fixé fermé.

2)la règle du plus petit Y modifié.

Le dépôt libre possédant le plus petit $Ymod_j$ est choisi et le dépôt j est tout d'abord fixé ouvert.

3.2.4.3 – Règles Delta

Ces règles sont adaptées de celles de Khumawala, utilisées dans la résolution du problème de localisation simple. Dans ces règles un ∇_{ji} était calculé, représentant la diminution minimale des coûts de transport pour le client i lorsque le dépôt j était fixé ouvert. Cette diminution était minimale puisque le coût de transport du dépôt j au client i était comparé au meilleur coût possible pour servir i par un autre dépôt ouvert ou pouvant le devenir (c'est-à-dire libre). Notons que dans le modèle de Khumawala, les X_{ji} représentent la fraction de la demande du client i satisfaite par le dépôt j. Nous savons cependant que le problème de localisation simple possède la propriété d'affectation unique, donc, dans une solution optimale, les X_{ji} sont 0 ou 1. Donc dans ces simplifications, les coûts impliqués dans les calculs sont sur la demande totale du client i.

Nous allons effectuer ces calculs pour le problème de localisation avec équilibrage. Ces calculs doivent cependant être effectués lorsque le client envoie son

offre vers les dépôts et lorsque qu'il reçoit sa demande des dépôts. Ce calcul ne représente cependant pas une borne inférieure sur la diminution des coûts de transport engendrée par l'ouverture d'un dépôt. Ceci puisque le problème de localisation avec équilibrage ne possède pas la propriété d'affectation unique. De plus, les arcs inter-dépôts ne sont pas utilisés dans le calcul, et ainsi, le calcul sur la diminution des coûts de transport est incomplet.

Mathématiquement:

$$\nabla_{ijp} = \min_{k \in (K_1 \cup K_2), k \neq j} \{\max\{(c_{kip} D_{ip} - c_{jip} D_{ip}) + (c_{ikp} O_{ip} - c_{ijp} O_{ip}), 0\}\}$$

$$\text{et } \Delta_j = \{\sum_{i \in C} \sum_{p \in P} \nabla_{ijp}\} - f_j$$

Deux règles peuvent être déduites:

1)la règle du plus grand Delta.

Le dépôt libre possédant le plus grand Delta est choisi et le dépôt j est tout d'abord fixé ouvert.

2)la règle du plus petit Delta.

Le dépôt libre possédant le plus petit Delta est choisi et le dépôt j est tout d'abord fixé fermé.

3.2.4.4 – Règles Oméga

Ces règles sont dérivées des règles Oméga utilisées par Khumawala dans la résolution du problème de localisation simple. Ces règles calculaient une borne supérieure sur la diminution des coûts de transport engendrée par l'ouverture d'un

dépôt j. Pour calculer cette borne dans le problème de localisation simple, le même calcul que celui des règles Delta est utilisé. Cependant les comparaisons sont faites seulement sur les dépôts ouverts. Dans le problème de localisation simple, ceci équivaut à calculer une telle borne supérieure puisque un maximum de flot va passer sur un dépôt j lorsqu'un minimum de dépôts sont ouverts.

Pour le problème de localisation avec équilibrage, le même calcul est effectué tout en tenant compte du fait que les clients ont une offre et une demande.

Les mêmes remarques s'appliquent à ces règles qu'aux règles Delta. D'abord puisque le problème de localisation avec équilibrage ne possède pas la propriété d'affectation unique, ce calcul ne peut pas vraiment être considéré comme une borne. De plus, puisque les arcs inter-dépôts ne sont pas considérés dans le calcul, nous ne tenons pas compte de tout le flot passant par un dépôt.

Mathématiquement:

$$\omega_{ijp} = \min_{k \in K_i} \{\max\{(c_{kip}D_{ip} - c_{jip}D_{ip}) + (c_{ikp}O_{ip} - c_{ijp}O_{ip}), 0\}\}$$

$$\text{et} \Omega_j = \{\sum_{i \in C} \sum_{p \in P} \omega_{ijp}\} - f_j$$

Deux règles peuvent être déduites:

1)la règle du plus grand Omega.

Le dépôt libre possédant le plus grand Omega est choisi et le dépôt j est tout d'abord fixé ouvert.

2)la règle du plus petit Omega.

Le dépôt libre possédant le plus petit Omega est choisi et le dépôt j est

tout d'abord fixé fermé.

3.2.4.5 – Règles du Quotient

Ces règles sont basées sur deux faits contradictoires. D'abord si la somme des coûts de transport sur les arcs inter-dépôts entrant ou sortant du dépôt j est élevée, alors le dépôt a une plus grande probabilité d'être fermé dans la meilleure solution entière du sous-arbre résultant. Par contre, si la somme des flots de tous les clients vers le dépôt j et des flots du dépôt j vers tous les clients est élevée, alors puisque le dépôt reçoit et retourne beaucoup de flot des clients, ce dépôt possède une plus grande probabilité d'être ouvert dans la meilleure solution entière du sous-arbre résultant.

Mathématiquement:

$$Q_j = \{\sum_{k \in D} \sum_{p \in P} (w'_{kjp} s_{kjp} + w'_{jkp} s_{jkp})\} / \{\sum_{i \in C} \sum_{p \in P} (x'_{ijp} + x'_{jip})$$

Deux règles peuvent être déduites:

1)la règle du plus grand Quotient.

Le dépôt libre possédant le plus grand Quotient est choisi et le dépôt j est tout d'abord fixé fermé.

2)la règle du plus petit Quotient.

Le dépôt libre possédant le plus petit Quotient est choisi et le dépôt j est tout d'abord fixé ouvert.

3.2.4.6 – Règles du Quotient modifiées

Ces règles sont semblables à celles du Quotient. Il s'agit simplement d'ajouter au numérateur des règles précédentes, le coût d'ouverture du dépôt (f_j). Intuitive-

ment lorsque ce coût est élevé, un dépôt possède une plus grande probabilité d'être fermé dans la meilleure solution entière du sous-arbre résultant. De même si ce coût est faible, un dépôt possède une plus grande probabilité d'être ouverte dans la meilleure solution entière du sous-arbre résultant.

Mathématiquement:

$$N_j = \{\{\sum_{k \in D} \sum_{p \in P} (w'_{kjp} s_{kjp} + w'_{jkp} s_{jkp})\} + f_j\} / \{\sum_{i \in C} \sum_{p \in P} (x'_{ijp} + x'_{jip})$$

Deux règles peuvent être déduites:

1)la règle du plus grand Quotient modifié.

Le dépôt libre possédant le plus grand Quotient modifié est choisi et le dépôt j est tout d'abord fixé fermé.

2)la règle du plus petit Quotient modifié.

Le dépôt libre possédant le plus petit Quotient modifié est choisi et le dépôt j est tout d'abord fixé ouvert.

3.2.4.7 – Règles Z

Ces règles sont semblables aux deux séries de règles précédentes. Dans cette série de règles, nous considérerons les coûts de transport totaux des flots entrant et sortant du dépôt, sur le dénominateur. Dans les deux règles précédentes nous ne considérions que les coûts de transport sur les arcs inter-dépôts entrant ou sortant du dépôt considéré.

Mathématiquement:

$$Z_j = \{\{\sum_{k\in D}\sum_{p\in P}(w'_{kjp}s_{kjp} + w'_{jkp}s_{jkp})\} + f_j +$$

$$\sum_{i\in C}\sum_{p\in P}(c_{ijp}x'_{ijp} + c_{jip}x'_{jip})\}/\{\sum_{i\in C}\sum_{p\in P}(x'_{ijp} + x'_{jip})$$

Deux règles peuvent être déduites:

1)la règle du plus grand Z.

Le dépôt libre possédant le plus grand Z est choisi et le dépôt j est tout d'abord fixé fermé.

2)la règle du plus petit Z.

Le dépôt libre possédant le plus petit Z est choisi et le dépôt j est tout d'abord fixé ouvert.

3.3 – Conclusion

Dans ce chapitre, nous avons étudié une méthode de "branch-and-bound" pour le problème de localisation avec équilibrage. Cette méthode utilise une relaxation faible pour le calcul des bornes inférieures à chaque noeud, puisque le modèle mathématique correspond à un problème de multiflot à coût minimum, facile à résoudre.

Nous avons également défini des méthodes de sélection de variables, ce qui, nous l'espérons, nous permettra de réduire considérablement le nombre de noeuds à explorer.

CHAPITRE 4 – LES RESULTATS

Nous présentons dans cette section les résultats obtenus sur les différentes méthodes de sélection de variables, ainsi que les performances générales de l'algorithme du "branch-and-bound" sur la relaxation faible du problème de localisation avec équilibrage. Nous comparons également l'efficacité de notre modèle à l'aide de données réelles. Tous nos tests ont été effectués sur un micro-ordinateur compatible PC AT muni d'une carte Definicon DSI-32 offrant un coprocesseur 32 bits avec une mémoire de deux méga-octets.

4.1 – Problèmes tests

Nous avons généré aléatoirement 72 problèmes tests à partir d'un réseau comprenant 263 villes du Mexique, des Etats-Unis et du Canada. Trois tailles de problème ont été générées, les *petits* possédant 50 noeuds, les *moyens* avec 150 noeuds et les *grands* possédant 263 noeuds. Dans chaque problème environ un noeud sur six est choisi pour être un dépôt, les autres deviennent des clients. Les problèmes possèdent également entre un et dix produits. Un coefficient de déséquilibre est également utilisé dans chaque problème. Ce coefficient est utilisé pour varier l'utilisation des arcs inter-dépôts.

Puisque dans la génération des arcs du réseau, les arcs inter-dépôts reçoivent des coûts plus bas que les autres arcs, généralement les clients reçoivent et expédient leurs marchandises à des dépôts rapprochés. En regroupant les noeuds du réseau en

différentes zones, il est raisonnable de supposer que les clients reçoivent et expédient leurs marchandises à des dépôts de la même zone. Si dans chaque zone une forte différence existe entre la demande totale et l'offre totale d'un produit, alors cette différence doit provenir d'une autre zone. Etant donné les coûts plus bas des arcs inter-dépôts, les transferts entre zones seront souvent effectués sur de tels arcs. Ainsi plus le nombre de zones est élevé, plus l'utilisation des arcs inter-dépôts est importante. Le nombre de zone de chaque problème constitue donc le coefficient de déséquilibre.

Chaque problème généré possède un code particulier, d'abord composé de deux lettres, d'abord une donnant la taille du problème (P, M, G) suivie d'un 'Z' et du nombre de zones contenues dans le problème. La lettre 'P' suivie du nombre de produits complète le code. Par exemple 'MZ1P7' correspond au problème de taille moyenne contenant une zone et 7 produits. Le tableau 1 donne les caractéristiques des problèmes pour chaque taille.

CARACTERISTIQUES	PETITS	MOYENS	GRANDS
nb. noeuds	50	150	263
nb. produits	1 à 10	1 à 10	1 à 4
nb. zones	1, 2 ou 3	1, 2 ou 4	1, 3 ou 5

Tableau 1 Caractéristiques des problèmes générés

Un problème est généré pour chaque combinaison possible des caractéristiques.

Ceci correspond à 30 petits et 30 moyens, mais seulement à 12 grands problèmes puisque la taille du fichier d'un grand problème est considérable pour plus de 4 produits.

Dans ce mémoire étant donné les restrictions sur l'utilisation d'un ordinateur, nos tests sont effectués sur les problèmes moyens contenant six produits ou moins. Les petits problèmes sont moins intéressants et les grands et les autres moyens sont trop long à résoudre. Ils ont été mis de coté pour des tests ultérieurs.

Nous utilisons d'abord les trois problèmes moyens contenant cinq produits pour déterminer l'efficacité des méthodes de sélections de variables. Nous obtenons alors une nouvelle méthode de sélection qui est testée sur les 18 problèmes moyens de six produits ou moins.

Finalement des données réelles provenant d'une compagnie de transport par conteneurs sont utilisées. Ce problème contient 130 dépôts, 289 clients et 12 produits. Dans le cas de ce problème nous voulons tester les performances de l'algorithme, mais surtout l'amélioration théorique possible sur les coûts d'opération de la compagnie en utilisant notre modèle mathématique. Cette amélioration est théorique puisque notre modèle se situe à un niveau tactique et non pas au niveau des opérations journalières.

4.2 - Implantation

Nous avons programmé en FORTRAN un algorithme de "branch-and-bound". Le programme fonctionne à l'aide d'un fichier de paramètres. Il a été développé de

façon à ce qu'il soit facile de remplacer les méthodes de sélection et le calcul de la borne inférieure à chaque noeud.

Pour nos tests nous utilisons une relaxation faible qui permet une décomposition en divers sous-problèmes de flot à coût minimum sans capacité. De tels problèmes sont résolus à l'aide du progiciel RNET. La procédure de calculs de la borne inférieure devient alors une transformation de nos données en des données utilisables par RNET.

4.3 – Méthodes de sélection de variables

Pour tester chacune des méthodes de sélection de variables, nous avons utilisé les trois problèmes de taille moyenne contenant 5 produits.

Pour chaque essai, nous nous sommes limités à mille noeuds explorés dans l'arbre du "branch-and-bound", puisque pour un problème de cette taille il existe un nombre de noeuds plus grand. Ce nombre de noeuds devrait être suffisant puisque dans l'algorithme, il est important de réduire rapidement la borne supérieure, ce qui est obtenu si une bonne solution entière est trouvée rapidement. Donc si une telle solution ne peut être trouvée dans les mille premiers noeuds explorés, la méthode ne sera pas efficace.

Nous retenons comme critères de comparaisons des différentes méthodes: l'éloignement de la meilleure solution obtenue par rapport à la borne inférieure, le noeud où cette solution est atteinte et le temps total d'exécution pour les mille itérations.

	Problèmes tests		
Règles	MZ1P5	MZ2P5	MZ4P5
borne inf.	121 134 608	106 251 744	131 116 016
Règle Y	130 097 189 7,4 183 13,6	115 199 416 8,4 9 13,2	139 188 128 6,2 802 12,7
Y modifié min. fixé à 1	130 362 592 7,6 711 n.d.	115 165 904 8,4 866 19,0	139 312 400 6,3 455 n.d.
Y modifié max. fixé à 0	130 829 480 8,0 764 14,4	* * * 14,0	140 598 080 7,2 947 15,2
Delta min. fixé à 0	131 480 152 8,5 792 21,6	* * * n.d.	140 625 712 7,3 969 18,8
Delta max. fixé à 1	130 362 592 7,6 758 20,6	115 427 176 8,6 933 19,5	139 312 400 6,3 559 18,4
Oméga min. fixé à 0	130 829 480 8,0 764 n.d.	* * * 14,2	141 350 144 7,8 3 15,3
Oméga max. fixé à 1	130 362 592 7,6 758 20,2	115 427 176 8,6 933 19,2	139 312 400 6,3 559 n.d.
Quotient min. fixé à 1	130 526 656 7,8 654 20,3	* * * n.d.	140 341 744 7,0 886 18,1
Quotient max. fixé à 0	* * * 16,5	* * * 14,9	141 230 560 7,7 2 15,0
Quotient modif. min. fixé à 1	129 907 152 7,2 391 n.d.	114 994 952 8,2 812 19,3	139 387 504 6,3 800 18,1
Quotient modif. max. fixé à 0	131 346 776 8,4 5 14,5	116 314 888 9,5 552 13,6	140 276 304 7,0 4 14,6
Z min. fixé à 1	130 108 824 7,4 879 n.d.	115 739 192 8,9 818 19,4	139 473 376 6,4 664 18,4
Z max. fixé à 0	131 467 816 8,5 3 15,8	116 549 352 9,7 n.d. n.d.	140 318 768 7,0 5 14,9

Tableau 2 Comparaisons entre les méthodes de sélection

Ces trois problèmes ont tout d'abord été testés en fixant la borne supérieure à 5% de la borne inférieure. Dans ce cas, l'arbre a été complètement exploré sans trouver de solution entière, ceci pour les trois problèmes tests. Nous avons alors ramené la borne supérieure initiale à 10% de la borne inférieure.

Dans le tableau 2, nous retrouvons d'abord la borne inférieure (c'est-à-dire la borne inférieure calculée au premier noeud du "branch-and-bound") de chaque problème testé, cette borne est la même peu importe la méthode utilisée. Nous présentons ensuite pour chaque méthode et chaque problème dans la ligne supérieure la meilleure solution entière obtenue et son écart par rapport à la borne inférieure. La ligne inférieure indique le noeud à laquelle la solution a été trouvée et le temps d'exécution par itération. Une étoile indique qu'aucune solution à moins de 10% de la borne inférieure n'a été trouvée en mille itérations.

Généralement, les méthodes qui fixent d'abord les variables à 1 trouvent une meilleure solution, cependant celles fixant d'abord à 0 utilisent moins de temps. Ceci est dû au fait qu'en fixant une variable à 0, la taille du problème est diminuée puisque les arcs rattachés à un dépôt fermé peuvent être éliminés.

Les deux meilleures méthodes sont respectivement la méthode du quotient modifié fixant d'abord à 1, qui a obtenue la meilleure solution pour deux des problèmes, et la méthode du Y qui a été la meilleure pour le troisième.

La règle du Y possède cependant l'avantage d'être plus rapide que les autres règles. Ceci en raison de la simplicité de son calcul, mais également puisqu'elle fixe presque toujours à 0.

En effet par la proposition 3.5, nous savons que la somme des y_j pour les j

libres, est inférieure ou égale à 1. Puisque les y_j sont contraints à être non-négatifs. S'il existe un y_j supérieur à 0.5 alors tous les autres seront inférieurs à 0.5 et donc plus près de 0 que de 1. Ainsi cette règle ne peut brancher plus d'une variable à 1.

Il semble donc qu'il est d'abord avantageux de fixer un certain nombre de variables à 0 puisque pour les deux premiers problèmes, la meilleure solution obtenue par la méthode Y est atteinte rapidement (183 et 9ème itération). Par contre les méthodes fixant d'abord à 1 ont généralement plus de succès.

Nous allons donc combiner la règle du Y pour son efficacité en temps de résolution et sa rapidité à atteindre une bonne borne supérieure, et la règle du quotient modifié fixant d'abord à 1 pour son efficacité à trouver la meilleure solution.

La nouvelle règle obtenue est la suivante:

Si $\min_{j \in K_2} \{(1 - y_j), y_j\} \leq \epsilon$

alors si $y_j \leq \epsilon$ (brancher d'abord y_j à 0)

si $(1 - y_j) \leq \epsilon$ (brancher d'abord y_j à 1)

sinon utiliser la règle du quotient modifié fixant d'abord à 1.

Le $\epsilon = 1/(2m)$ où m représente le nombre de candidats-dépôts. Ce calcul représente la moitié de la moyenne des y_j, ceci étant donné le résultat de la proposition 3.5.

4.4 – Tests sur la nouvelle règle

Nous avons testé cette nouvelle règle sur les 18 problèmes de taille moyenne

Produits	Problèmes					
	MZ1P1		MZ2P1		MZ4P1	
1	37 382 464		7 938 558		15 013 881	
	42 319 824	13,2	n.d.		n.d.	
	42 540 716	13,8	*	*	*	*
	650	3,1	*	3,5	*	3,7
	MZ1P2		MZ2P2		MZ4P2	
2	16 191 384		27 610 428		27 966 322	
	n.d.		33 493 498	21,3	n.d.	
	*	*	*	*	*	*
	*	5,4	*	5,1	*	5,9
	MZ1P3		MZ2P3		MZ4P3	
3	52 921 420		63 335 508		83 647 448	
	59 865 960	13,1	68 776 528	8,6	n.d.	
	60 230 304	13,8	69 691 624	10,0	91 675 064	9,6
	98	8,5	531	n.d.	308	8,7
	MZ1P4		MZ2P4		MZ4P4	
4	101 381 736		107 880 448		84 571 432	
	n.d.		115 827 424	7,4	n.d.	
	109 040 640	7,6	115 970 744	7,5	94 427 712	11,7
	784	10,8	n.d.	11,7	50	11,7
	MZ1P5		MZ2P5		MZ4P5	
5	121 134 608		106 251 744		131 116 016	
	129 038 768	6,5	113 963 760	7,3	n.d.	
	129 603 600	7,0	115 061 616	8,3	139 104 064	6,1
	713	15,0	233	14,7	795	n.d.
	MZ1P6		MZ2P6		MZ4P6	
6	245 390 256		239 504 544		203 243 728	
	252 917 120	3,1	n.d.		n.d.	
	253 521 440	3,3	248 280 800	3,7	214 089 600	5,3
	924	21,2	9	15,7	66	18,2

Tableau 3 Résultats sur la nouvelle méthode

comprenant entre 1 et 6 produits (3 de chaque catégorie). En particulier les trois problèmes contenant 5 produits sont les mêmes que ceux utilisés dans la section précédente. Nous avons fixé la borne supérieure initiale à 20% de la borne inférieure et nous avons stoppé l'algorithme après 1000 itérations.

Dans le tableau 3 pour chaque problème testé, nous avons d'abord son code, suivi de sa borne inférieure calculée au premier noeud du "branch-and-bound" et d'une seconde borne inférieure calculée à partir d'un algorithme dual (voir GRAR (1988)) avec son écart en pourcentage de la première. La quatrième ligne contient la meilleure solution trouvée et son écart en pourcentage avec la première borne inférieure. La dernière ligne contient l'itération à laquelle la solution a été découverte et le temps moyen (en secondes) par noeud.

Pour les huit problèmes où la borne inférieure de l'algorithme dual était connue, cette borne est proche de notre meilleure solution. Le plus grand écart apparaissant pour MZ2P3 avec une borne inférieure duale à 8,6% contre une solution à 10,0% de la borne inférieure initiale. Dans les autres cas la différence entre la borne duale et la meilleure solution est toujours à 1,0% ou moins.

Nous avons poussé nos tests plus loin en tentant de calculer la véritable solution optimale pour le problème MZ1P1, en explorant complètement l'arbre du "branch-and-bound". Ce problème a été choisi parce que parmi ceux pour lesquels une solution a été trouvée, il possédait le plus petit temps d'exécution.

Puisque nous avons une solution réalisable à 13,8%, nous fixons la borne supérieure initiale à 13,9% de façon à accélérer le plus possible l'algorithme. Après 52139 itérations (près de deux jours de temps machine), le programme a été inter-

rompu sans que l'arbre ait pu être exploré complètement. Une solution réalisable légèrement meilleure que celle déjà connue a été découverte. Le tableau 4 résume les résultats de cette expérience.

	Valeur	Ecart	Itération
Borne inférieure	37 382 464		
Borne inf. duale	42 319 824	13,2	
Meilleure solution	42 498 780	13,7	9264
Première amélioration	42 539 880	13,8	9261
Ancienne solution	42 540 716	13,8	

Tableau 4 Autres résultats sur MZ1P1

Ce tableau contient la borne inférieure initiale de MZ1P1 et celle calculée par l'algorithme dual. La meilleure solution trouvée et la première amélioration sur l'ancienne solution. Ces deux nouvelles solutions sont présentées avec leur écart de la borne inférieure et l'itération à laquelle elles ont été trouvées. Finalement, l'ancienne solution, celle trouvée dans les 1000 premières itérations, est présentée.

Il faut plus de 9000 itérations pour améliorer notre solution optimale, malgré la réduction de la borne supérieure initiale. Cette amélioration n'est que de 0,1% de la borne inférieure initiale. Après ces amélioration très rapprochées aucune autre amélioration ne survient pour les quelques 43 000 itérations suivantes. Notre solution optimale trouvée dans les 1000 premières itérations est donc difficile à

améliorer et de plus elle est près de la borne inférieure duale.

	Problèmes tests		
Règles	MZ1P5	MZ2P5	MZ4P5
borne inférieure	121 134 608	106 251 744	131 116 016
borne inf. duale	129 038 768	113 963 760	n.d.
Règle Y	130 097 189 7,4 183 13,6	115 199 416 8,4 9 13,2	139 188 128 6,2 802 12,7
Quotient modif. min. fixé à 1	129 907 152 7,2 391 n.d.	114 994 952 8,2 812 19,3	139 387 504 6,3 800 18,1
Nouvelle règle	129 603 600 7,0 713 15,0	115 061 616 8,3 233 14,7	139 104 064 6,1 795 n.d.

Tableau 5 Comparaisons entre la nouvelle méthode et les autres retenues comme efficaces

Le tableau 5 nous permet de comparer la nouvelle méthode avec les règles Y et du quotient modifié, pour les problèmes moyens contenant 5 produits. Pour MZ1P5 et MZ4P5 la meilleure solution entière trouvée est légèrement meilleure que celle trouvée dans les comparaisons des différentes méthodes. Pour MZ2P5 la solution trouvée est à 8,3% de la borne inférieure contre 8,2% en utilisant la méthode du quotient modifié seule. Cependant le résultat le plus intéressant est la réduction significative du temps d'exécution par itération entre cette méthode et la méthode du quotient modifié seule. Par exemple pour MZ2P5, 4,6 secondes en moyenne sont épargnées par itération, ce qui représente 4600 secondes (plus d'une heure) pour 1000 itérations.

Cette nouvelle méthode constitue donc un compromis entre les règles Y et du quotient modifié. Pour les trois problèmes testés, elle a obtenue une meilleure solution que la règle Y et a été plus rapide que la règle du quotient modifié fixant d'abord à 1. De plus, pour deux de ces problèmes elle a trouvé une meilleure solution que la règle du quotient modifié. Ceci est en accord avec les raisons qui nous avaient conduit à cette nouvelle méthode. De plus lorsque la borne inférieure duale est disponible, notre solution optimale est toujours rapporchée de cette borne. Notre mlthode permet donc de trouver relativement rapidement sur micro-ordinateur une solution près de la solution optimale.

4.5 – Tests sur données réelles

Nous testons finalement notre algorithme avec des données réelles étalées sur une année et provenant d'une compagnie de transport par conteneurs. Le but de ces tests n'est pas de tester les méthodes de sélection des variables à fixer, mais plutôt de tester l'efficacité du modèle mathématique du problème de localisation avec équilibrage. Nous comparons donc sommairement les résultats obtenus avec notre modèle aux opérations de la compagnie. (Pour une étude plus complète sur ces données voir Dejax et coll. (1987)).

Le tableau 6 compare les opérations de la compagnie (résultats observés) aux résultats calculés avec notre algorithme. Les coûts calculés par notre algorithme sont beaucoup plus bas (41%) que ceux observés, malgré le fait que la solution calculée demande plus de mouvements (véhicules transportés). Cependant il faut

noter que notre méthode se situe à un niveau tactique par opposition aux opérations de la compagnie qui se situe à un niveau au jour le jour. Il est alors diffcile de toujours choisir le meilleur endroit pour envoyer un conteneur lorsqu'une décision instantanée doit être prise. La différence entre les deux solutions doit alors être vue comme étant une borne supérieure sur l'amélioration possible des opérations de la compagnie.

	Résultats observés	Résultats calculés
coûts totaux	87 788 267	51 730 908
véhicules transportés	87 599	94 334
dépôts utilisés	87	48

Tableau 6 Tests sur données réelles

Le résultat le plus intéressant provient du fait que seulement 48 dépôts ont été utilisés dans la solution calculée contre 87 dans la solution observée. Il semble donc qu'une réduction du nombre de dépôts utilisés soit la base pour une meilleure gestion des opérations de la compagnie.

CONCLUSION

Nous avons développé un algorithme de "branch-and-bound" pour le problème de localisation avec transferts entre les dépôts en utilisant une relaxation faible pour le calcul de la borne inférieure à chaque noeud. Nous avons également développé des méthodes de sélection des variables à fixer qui permettent d'accélérer la convergence.

Nos tests nous ont permis de trouver une autre méthode de sélection des variables à fixer en combinant les deux meilleures parmi celles déjà définies. Une autre série de tests sur cette nouvelle méthode nous a alors permis de montrer l'efficacité de l'algorithme en comparant la solution optimale obtenue avec une borne inférieure calculée par un algorithme dual. Pour les problèmes où cette borne inférieure était disponible, notre meilleure solution était toujours proche de cette borne, malgré le fait que nous limitons à 1000 le nombre de noeuds explorés dans le "branch-and-bound".

Nous avons également utilisé des données réelles. En comparant nos résultats avec les opérations de la compagnie, nous avons obtenue une différence significative de 41%. Cette différence est cependant théorique et représente en fait une borne supérieure sur l'amélioration possible sur les opérations de la compagnie. Ceci puisque les opérations de la compagnie sont effectuées au jour le jour par opposition à notre modèle qui se situe à un niveau tactique. Il est intéressant de noter que notre algorithme utilise 39 dépôts de moins que la compagnie. Une meilleure gestion des dépôts est certainement nécessaire pour obtenir une réduction des coûts d'opération. Ω Le développement d'autres méthodes encore plus efficaces pourrait

accélérer la convergence de l'algorithme, mais puisque nos méthodes conduisent déjà à une solution près de l'optimum, ces amélioration ne seraient que légères. L'amélioration la plus importante serait de développer une méthode permettant de résoudre efficacement la relaxation forte du problème. A ce moment, les bornes inférieures calculées à chaque noeud seraient plus précises et donc le nombre de noeuds pourrait être de beaucoup diminué.

REFERENCES

CORNUEJOLS, G., NEMHAUSER, G.L. et WOLSEY, L.A. (1983). "The Uncapacitated Facility Location Problem", *Management Science Research Report No. MSRR 498*, Pittsburgh, Pennsylvanie, 73 p.

CRAINIC, T.G., DEJAX, P. et DELORME, L. (1986). "Problèmes de localisation-distribution avec équilibrage : éléments de modélisation et propriétés". *Rapport 490, Centre de Recherche sur les Transports, Université de Montréal*, 35 p.

DEJAX, P., CRAINIC, T.G., DELORME, L., BLOISE, M., de TOCQUEVILLE, E. et HODGSON, J. (1987). "Planification tactique du transport terrestre des conteneurs vides, note de synthèse". *Rapport de recherche pour la COMPAGNIE GENERALE MARITIME. Ecole Centrale de Paris et Université de Montréal.*

DEJAX, P., CRAINIC, T.G., DELORME, L., BLOISE, M., de TOCQUEVILLE, E. et HODGSON, J. (1987). "Planification tactique du transport terrestre des conteneurs vides, Rapport d'annexe". *Rapport de recherche pour la COMPAGNIE GENERALE MARITIME. Ecole Centrale de Paris et Université de Montréal.*

EFROYMSON, M.A. et RAY, T.L. (1966). "A Branch and Bound Algorithm for Plant Location", *Operations Research 14*, 361-368.

GRAR, F. (1988). "Méthodes duales pour la résolution d'un problème de localisation avec transferts entre les dépôts", *Mémoire de maîtrise, Université de Montréal*, en préparation.

KHUMAWALA, B.M. (1972). "An Efficient Branch and Bound Algorithm for the Warehouse Location Problem", *Management Science 18B*, 718–731.

KRARUP, J. et PRUZAN, P.M. (1983). "The Simple Plant Location Problem : Survey and Synthesis", *European Journal of Operational Research 12*, 36–81.

SALKIN, Harvey M. (1975). *Integer Programming*, Addison-Wesley, Reading, Massachusetts.

ANNEXE A

Guide d'utilisation du programme de "branch-and-bound" sur la localisation avec transferts entre les dépôts.

Nous présentons dans cette annexe les éléments nécessaires au fonctionnement du programme que nous avons conçu pour nos tests. Ce programme est codé en Fortran, il utilise le logiciel RNET lorsqu'il est nécessaire de résoudre un problème de flot à coût minimum dans l'algorithme. Les éléments nécessaires sont un fichier de données, la préparation de COMMON et un fichier de paramètres.

A.1 – Fichier de données

Le fichier est composé des 5 sections suivantes, qui doivent apparaître dans l'ordre.

1 – les candidats-dépôts et leur coût fixe d'ouverture.

2 – les noms des clients.

3 – les noms des modes.

4 – les offres et les demandes.

5 – le réseau.

Les noms des candidats-dépôts, des clients, des modes et des produits sont représentés par des champs de 4 caractères (format A4 en Fortran). Les coûts fixes d'ouverture des candidats-dépôts et les coûts des arcs du réseau sont représentés par des champs de 10 caractères, contenant 2 décimales (format F10.2). Finalement les quantités d'offres et de demandes sont représentées par des champs de 9 caractères selon le format F9.0.

A.1.1 – Les candidats-dépôts

Chaque candidat-dépôt apparaît sur un enregistrement différent avec son nom suivi de son coût fixe d'ouverture. Les enregistrements doivent être ordonnés par ordre croissant du nom. Le format d'un enregistrement est:

champ:

1 à 4 : code de 4 caractères représentant le candidat-dépôt.

6 à 15 : coût d'ouverture du candidat-dépôt.

Le dernier enregistrement de cette section contient le code 'ZZZZ' dans les positions 1 à 4.

A.1.2 – Les clients

Chaque client apparaît sur un enregistrement différent avec son identification (nom). Les enregistrements doivent être ordonnés par ordre croissant du nom. Le nom du client apparaît dans les caractères 1 à 4 de l'enregistrement.

La fin de cette section est représentée par un enregistrement contenant 'ZZZZ' dans les caractères 1 à 4.

A.1.3 – Les modes

Chaque mode apparaît sur un enregistrement différent. Les codes des modes doivent être ordonnés par ordre croissant. Les caractères 1 à 4 d'un enregistrement contiennent le nom du mode.

La fin de cette section est représentée par un enregistrement contenant 'ZZZZ'

dans les caractères 1 à 4.

A.1.4 – Les offres et les demandes

Un enregistrement de cette section contient l'offre et la demande associées à un produit et un client. Les enregistrements sont classés d'abord par ordre croissant du code de produit, puis par ordre croissant du code de client. Notons que nous profitons de cette section pour identifier les produits, puisqu'aucune section spécifique n'est prévue pour les définir. Le format d'un enregistrement est:

champ :

1 à 4 : code de 4 caractères représentant le produit.

6 à 9 : code de 4 caractères représentant le client.

10 à 18: offre du client pour le produit.(format F9.0)

19 à 27: demande du client pour le produit.(format F9.0)

Le dernier enregistrement de cette section contient le code 'ZZZZ' dans les positions 1 à 4.

A.1.5 – Le réseau

Un enregistrement de cette section spécifie les caractéristiques d'un arc du réseau. Le format est le suivant:

champ :

1 à 4 : code de 4 caractères représentant le noeud origine de l'arc (ce noeud peut être un dépôt ou un client).

6 à 9 : code de 4 caractères représentant le noeud destination de l'arc (ce noeud peut être un dépôt ou un client).

11 à 14: code de 4 caractères représentant le mode de l'arc.

16 à 19: code de 4 caractères représentant le produit de l'arc.

20 à 29: coût de l'arc selon le format F10.2.

Les enregistrements doivent être ordonnés de cette façon:

- Pour chaque dépôt par ordre croissant du code, nous avons:
  - Tous les arcs allant vers d'autres dépôts.
  - Le code 'ZZZZ' dans les positions 1 à 4 indiquant la fin de cette sous-section.
  - Tous les arcs allant vers les clients.
  - Le code 'ZZZZ ZZZZ' dans les positions 1 à 9, indiquant la fin de cette sous-section.

- Pour chaque client par ordre croissant du code, nous avons:
  - Tous les arcs allant vers les dépôts.
  - Le code 'ZZZZ ZZZZ' dans les positions 1 à 9, indiquant la fin de cette sous-section.

- Dans chaque sous-section, les enregistrements sont classés:
  - Par ordre croissant du code du noeud destination (soit un client, soit un dépôt, mais jamais les deux dans une même section).
  - Par ordre croissant du code de mode.
  - Par ordre croissant du code de produit.

A.2 – Préparation des COMMON

Le programme nécessite la préparation de trois fichiers de COMMON. Ces fichiers sont automatiquement inclus dans le programme principal. Ces trois fichiers sont sur la première page de l'annexe suivante et sont appelés dans l'ordre COMPILE.CMN, COMLOC2.CMN et COMPILE.CMN.

Dans chacun de ces COMMON, certaines dimensions de tableaux ont été remplacées par des lettres. Ces lettres doivent être remplacées par les valeurs suivantes:

A : nombre maximal d'arcs.

N : nombre maximal de clients.

M: nombre maximal de candidats-dépôts.

P : nombre maximal de produits.

D : nombre maximal de modes.

Ces valeurs doivent également être placées dans le programme. Les variables à modifier se trouvent dans la procédure INIT du programme. Voici les variables à modifier et leurs significations:

MAXARC : nombre maximal d'arcs.

MAXCLT : nombre maximal de clients.

MAXDPT : nombre maximal de candidats-dépôts.

MAXNOD: MAXCLT + MAXDPT.

MAXPRD : nombre maximal de produits.

MAXMOD: nombre maximal de modes.

A.3 - Fichier de paramètres

Lorsque l'exécution du programme débute, un nom de fichier de paramètres est demandé interactivement à l'utilisateur. Ce fichier permet de donner les renseignements nécessaires sur le problème à traiter. Il donne par exemple les noms des fichiers de données et de résultats, et permet également de modifier certains paramètres prédéfinis.

Le fichier peut contenir plusieurs problèmes à traiter à la suite les uns des autres. Le fichier permet alors de modifier toutes ou seulement une partie des données du problème précédent.

Un fichier de paramètres contient des paragraphes qui eux-mêmes contiennent des instructions. Ces paragraphes et ces instructions sont des chaînes de 6 caractères toujours placés en début de ligne. Les instructions peuvent être suivies de paramètres.

Les paragraphes sont tous facultatifs, sauf pour le premier d'une série de problèmes à traiter. Il doivent absolument être dans l'ordre dans lequel il sont présentés.

A.3.1 - Paragraphe 'OUTPUT'

Ce paragraphe est obligatoire pour le premier problème du fichier de paramètres. Ce paragraphe permet de définir le fichier des résultats.

La seule instruction pour ce paragraphe est 'FICRES' suivi d'un espace et du nom du fichier de résultats, comptant au maximum 40 caractères.

Un nouveau fichier de résultats peut être défini pour chaque nouveau problème. Dans le cas contraire, les résultats d'un second problème sont écrit à la suite du premier dans le même fichier.

A.3.2 – Paragraphe 'CTITRE'

Ce paragraphe facultatif est utilisé pour donner un titre au problème traité. La ligne suivant ce verbe doit contenir un titre dans les premiers 40 caractères.

A.3.3 – Paragraphe 'CINPUT'

Ce paragraphe permet de définir le ou les fichiers contenant les données du problème à traiter. Il est nécessaire pour le premier problème, mais facultatif pour les suivants, puisqu'un nouveau problème peut être obtenu du premier en modifiant seulement certaines données ou encore en changeant certaines constantes de l'algorithme.

Les instructions de ce paragraphe ont 6 caractères placés dans les espaces 1 à 6 de la ligne. Elles sont suivis d'un nom de fichier d'au maximum 40 caractères, contenu dans les espaces 8 à 47. Ces instructions sont:

NMFICH : Pour obtenir le nom du fichier contenant toutes les données du problème.

NDEPOT : Pour obtenir le nom du fichier contenant la section des noms et des coûts fixes d'ouverture des dépôts.

NCLIEN : Pour obtenir le nom du fichier contenant la section des noms des clients.

NMMODE: Pour obtenir le nom du fichier contenant la section des noms des modes.

NOFDEM : Pour obtenir le nom du fichier contenant la section des offres et des demandes.

NRESAU : Pour obtenir le nom du fichier contenant la section du réseau.

Si toutes les données sont dans un même fichier, sauf par exemple le réseau, il est possible d'utiliser NMFICH puis NRESAU qui va définir le réseau comme se trouvant ailleurs.

Il n'est pas à conseiller de trop séparer le fichier de données en plusieurs. L'utilité d'un fichier contenant une partie seulement du fichier global intervient lorsque par exemple seule une des sections du fichier doit être remplacée pour traiter un nouveau problème.

A.3.4 - Paragraphe 'MODINP'

Ce paragraphe facultatif est utilisé seulement lorsque certaines données du fichier doivent être modifiées. Par exemple pour augmenter de 10% les coûts sur certains arcs. Il est possible de modifier quatre type de données, soit les coûts fixes d'ouverture des dépôts, les offres, les demandes et les coûts de transport sur les arcs.

Toutes ces instructions demandent de préciser l'enregistrement à modifier. Si tous les enregistrements d'un certains type doivent être modifiés, le paramètre 'TOUS' peut remplacer un nom. Par exemple pour modifier tous les coûts fixes d'ouverture, au lieu de spécifier une instruction pour chaque dépôt, une seule in-

struction peut être spécifiée avec le terme 'TOUS'.

A.3.4.1 – Modification des coûts fixes d'ouverture

Les instructions FDEPOT, CDEPOT et ADEPOT sont utilisées pour modifier les coûts fixes d'ouverture des dépôts. Le format de chacune de ces instructions est: champ :

1 à 6 : code de 6 caractères de l'instruction.

8 à 11 : code de 4 caractères représentant le dépôt à modifier (TOUS peut être utilisé pour représenter tous les dépôts).

13 à 27: facteur de modification selon le format F15.5.

La fonction de ces instructions est la suivante:

FDEPOT: Pour multiplier le coût du dépôt par le facteur de modification.

CDEPOT: Pour remplacer le coût du dépôt par le facteur de modification.

ADEPOT: Pour ajouter au coût du dépôt le facteur de modification.

Une combinaison de ces instructions peut être utilisée sur le même dépôt dans le même problème. Par exemple pour ajouter un facteur à un coût, puis pour augmenter ce nouveau coût de 50%.

A.3.4.2 – Modification des offres et des demandes

Ces instructions sont peu utiles et sont très délicates. En effet pour que le problème soit réalisable la somme des offres doit être égale à celles des demandes pour chaque produit. Ainsi la modification d'une offre ou d'une demande peut rendre le problème non-réalisable. De plus pour pouvoir utiliser le logiciel RNET dans le programme, les offres et les demandes doivent être entières.

Il est quand même possible d'utiliser ces instructions par exemple pour doubler

toutes les offres et les demandes.

Ces instructions sont FOFFRE, COFFRE et AOFFRE pour modifier les offres et FDEMAN, CDEMAN et ADEMAN pour modifier les demandes. Le format d'une de ces instructions est:

champ :

1 à 6 : code de 6 caractères de l'instruction.

8 à 11 : code de 4 caractères représentant le dépôt à modifier (TOUS peut être utilisé pour représenter tous les dépôts).

13 à 16: code de 4 caractères représentant le produit à modifier (TOUS peut être utilisé pour représenter tous les produits).

18 à 32: facteur de modification selon le format F15.5.

La fonction de ces instructions est la suivante:

FOFFRE, FDEMAN: Pour multiplier l'offre ou la demande par le facteur de modification.

COFFRE, CDEMAN: Pour remplacer l'offre ou la demande par le facteur de modification.

AOFFRE, ADEMAN: Pour ajouter à l'offre ou à la demande le facteur de modification.

A.3.4.3 – Modification des coûts de transport des arcs

Cette catégorie d'instructions est utilisée pour modifier les coûts de transport sur un ou une catégorie d'arcs contenues dans le réseau. Ces instructions sont FCOUTS, CCOUTS et ACOUTS. Le format d'une telle instruction est:

champ :

1 à 6 : code de 6 caractères de l'instruction.

8 à 11 : code de 4 caractères représentant l'origine (client ou dépôt) de l'arc à modifier (DEPO peut être utilisé pour représenter tous les dépôts, CLNT pour représenter tous les clients ou TOUS pour représenter les deux).

13 à 16: code de 4 caractères représentant la destination (client ou dépôt) de l'arc à modifier (DEPO, CLNT et TOUS peuvent être utilisés).

18 à 21: code de 4 caractères représentant le mode de l'arc à modifier (TOUS peut être utilisé pour représenter tous les modes).

23 à 26: code de 4 caractères représentant le produit de l'arc à modifier (TOUS peut être utilisé pour représenter tous les produits).

27 à 41: facteur de modification selon le format F15.5.

La fonction de ces instructions est la suivante:

FCOUTS: Pour multiplier le coût de l'arc par le facteur de modification.

CCOUTS: Pour remplacer le coût de l'arc par le facteur de modification.

ACOUTS: Pour ajouter au coût de l'arc le facteur de modification.

Il est possible par exemple de modifier tous les arcs pour un produit donné en donnant le paramètre TOUS à l'origine, la destination et au mode.

A.3.5 – Paragraphe 'CNSTAN'

Ce paragraphe permet de modifier les valeurs de défaut de certaines constantes de l'algorithme, tel que le critère d'arrêt, le nombre maximal d'itérations, la méthode

de branchement, etc.

Les instructions sont les suivantes:

METHO1: Fait terminer l'exécution du programme lorsque la première solution entière inférieure à notre critère d'arrêt, est trouvée.

METHO2: Ne fait pas terminer l'exécution du programme lorsque la première solution entière inférieure à notre critère d'arrêt, est trouvée (Si aucune des deux méthodes n'est précisée, la méthode 1 est choisie).

CARRET: Contient le pourcentage d'écart à la borne inférieure, pour la borne supérieure initiale (1% comme valeur de défaut). Le pourcentage est contenu dans les espaces de 7 à 16 selon le format F10.8.

MXITER : Contient le nombre maximal d'itération au bout duquel l'algorithme s'arrête (valeur de défaut 1000). Cette valeur est contenue dans les espaces 7 à 16 selon le format I10.

METBRN: Contient la méthode de branchement utilisée. La méthode choisie est caractérisée par une chaîne d'au maximum 10 caractères (format A10) contenue dans les espaces 7 à 16. Les méthodes possibles sont:

: YRULE : représente la règle Y (règle de défaut).

: MODYRULMIN : règle Y modifiée fixant d'abord à 1.

: MODYRULMAX : règle Y modifiée fixant d'abord à 0.

: DELTAMIN : règle Delta fixant d'abord à 0.

: DELTAMAX : règle Delta fixant d'abord à 1.

: OMEGAMIN : règle Oméga fixant d'abord à 0.

: OMEGAMAX : règle Oméga fixant d'abord à 1.

: QUOTIENMIN : règle du Quotient fixant d'abord à 1.

: QUOTIENMAX : règle du Quotient fixant d'abord à 0.

: MODQUOTMIN : règle du Quotient modifié fixant d'abord à 1.

: MODQUOTMAX : règle du Quotient modifié fixant d'abord à 0.

: ZRULEMIN : règle Z fixant d'abord à 1.

: ZRULEMAX : règle Z fixant d'abord à 0.

Notons que lorsqu'un deuxième problème est contenu dans le fichier des paramètres, les valeurs fixées lors du premier problème deviennent les valeurs de défaut.

A.3.6 – Paragraphe 'SCTION'

Ce paragraphe ne contient aucune instruction. Il est utilisé simplement pour indiquer que tous les paramètres ont été lus pour le problème, mais qu'un autre problème suit.

A.3.6 – Paragraphe 'TERMIN'

Ce paragraphe ne contient aucune instruction. Il est utilisé pour indiquer que tous les problèmes du fichier de paramètres ont été lus. Ce paragraphe est donc toujours présent à la dernière ligne du fichier.